Ricci Curvature based Graph Sparsification for Continual Graph Representation Learning

Xikun Zhang, Dongjin Song*, *Member, IEEE*, and Dacheng Tao*, *Fellow, IEEE*

***Abstract*—Memory replay, which stores a subset of historical data from previous tasks to replay while learning new tasks, exhibits state-of-the-art performance for various continual learning applications on Euclidean data. While topological information plays a critical role in characterizing graph data, existing memory replay based graph learning techniques only store individual nodes for replay and do not consider their associated edge information. To this end, based on the message passing mechanism in GNNs, we present Ricci curvature based graph sparsification technique to perform continual graph representation learning. Specifically, we first develop the Subgraph Episodic Memory (SEM) to store the topological information in the form of computation subgraphs. Next, we sparsify the subgraphs such that they only contain the most informative structures (nodes and edges). The informativeness is evaluated with the Ricci curvature, a theoretically justified metric to estimate the contribution of neighbors to represent a target node. In this way, we can reduce the memory consumption of a computation subgraph from $\mathcal{O}(d^L)$ to $\mathcal{O}(1)$, and enable GNNs to fully utilize the most informative topological information for memory replay. Besides, to ensure the applicability on large graphs, we also provide the theoretically justified surrogate for the Ricci curvature in the sparsification process, which can greatly facilitate the computation. Finally, our empirical studies show that SEM outperforms state-of-the-art approaches significantly on four different public datasets. Unlike existing methods which mainly focus on task incremental learning (task-IL) setting, SEM also succeeds in the challenging class incremental learning (class-IL) setting in which model is required to distinguish all learned classes without task indicators, and even achieves comparable performance to joint training which is the performance upper bound for continual learning.**

***Index Terms*—Graph neural networks, Continual learning, Graph representation learning, Continual graph learning**

I. Introduction

In real-world graph applications, it is critical to ensure that Graph Neural Networks (GNNs) [1]–[3] are capable of continually adapting to new tasks without interfering with their performance over previous tasks. Because of this, continual graph representation learning is attracting increasingly more attention recently. For example, a community detection model is expected to detect the newly emerged communities in a social network while maintaining its capability to recognize existing communities; a document classifier should be able to classify articles belonging to either existing or newly emerged research areas in a citation network, *etc.* However, the rich topological connections among different data samples (graph nodes) pose great challenges to applying some most effective continual learning techniques on graph data.

Memory replay, inspired by research on cognitive science [4], [5], has demonstrated state-of-the-art performance in various classical continual learning tasks. The key idea is to replay a subset of data samples from previous tasks over the model while learning the new tasks [6]–[8]. Due to its success, memory replay is also adopted for continual learning on graph data by storing and replaying representative nodes [9], [10]. For graph data, however, only storing a subset of nodes for replay will neglect the important topological information. Since the properties of the target node are not only determined by itself but also by its neighbors, in this paper, we propose to store representative computation subgraphs to explicitly preserve the topological information for memory replay. Directly storing computation subgraphs, however, will trigger the memory explosion problem. Supposing the average node degree is d, then in a L-layer GNN following the message passing paradigm [2], the size of the neighboring nodes in the computation subgraph of a node will be $\mathcal{O}(d^L)$, let alone the associated edges which are typically several orders of magnitude more than the nodes. For instance, in the Reddit-CL dataset, the average node degree is 492, and the maximal degree is 21,657. Obviously, directly storing the entire computation subgraphs is infeasible.

To fully utilize the topological information while maintaining a tractable space complexity at the same time, we propose to sparsify the computation subgraph and preserve the most informative structures (nodes and edges) by utilizing the edge based Ricci curvature [11], [12]. Specifically, the Ricci curvature quantifies how easily information is propagated between two nodes. Given a pair of nodes u and v connected by an edge e_{uv}, the curvature Ric(u,v) is calculated based on the surrounding topological connections like the triangles and 4-cycles based at e_{uv}. Consequently, a higher curvature implies that information could be propagated more easily from u to v and vice versa (as shown in Figure 1(b)). In other words, among all edges connected to a node v, the ones with higher curvatures contribute more in generating the representation of v and should be preserved. In contrast, the edges with low curvatures provide little information to v and can be deleted without significant influence on the representation of v. Therefore, to sparsify the computation subgraph of a node v containing its L-hop neighbors, we propose to sample a subset of the edges and their associated nodes based on curvatures. To ensure the connectivity of sparsified subgraphs, we start

X. Zhang and D. Tao are with the Sydney AI Centre and the School of Computer Science in the Faculty of Engineering at The University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia (Email: xzha0505@uni.sydney.edu.au, dacheng.tao@gmail.com).

Dr. D. Song is with the Department of Computer Science and Engineering, University of Connecticut, Storrs, Connecticut, the United States. Email: dongjin.song@uconn.edu.
* indicates corresponding authors.

by sampling the 1-hop neighbors, and then iteratively sample the higher-order neighbors hop by hop. In this way, we can fix the budget for selected edges and nodes, and the size of the sparsified computation subgraph will be independent of the original computation subgraph, *i.e.*, the memory space complexity can be reduced from $\mathcal{O}(d^L)$ to $\mathcal{O}(1)$.

The Ricci curvature has several formulations. In this work, we adopt the Balanced Forman curvature [11] which is theoretically justified to be negatively correlated to the information bottleneck. Considering that the computation complexity of Balanced Forman curvature is $|\mathbb{E}|d^2_{\max}$, which could be time-consuming to compute on dense graphs, we circumvent computing the exact curvature values by approximating it with the graph diffusion process formulated as a lazy random walk. We theoretically demonstrate that under certain rules, the probability distribution at a node u of a lazy random walk starting at node v is positively correlated to the Ricci curvature between v and u. Therefore, in practice, we sample the edges based on the probability distribution of the random walk and use it as a surrogate for the exact Ricci curvature. Note that we are aware of existing graph sparsification techniques [13]–[22], however, most of them are not applicable for continual graph representation learning as detailed in Section II-B.

In the experiments, we compare Ricci curvature-based sampling with uniform sampling, degree-based sampling [23], and justify its effectiveness. Based on our thorough empirical studies on four different public datasets, SEM with Ricci curvature-based sampling outperforms the existing state-of-the-art methods, especially in the challenging class-IL scenario. Moreover, the performance of SEM is even comparable to joint training which is the performance upper bound for continual learning. To summarize, our contributions are:

1) We develop the Subgraph Episodic Memory (SEM) to store the explicit topological information in the form of computation subgraphs and perform memory replay based continual graph representation learning.
2) We resolve the memory explosion problem by sparsifying the subgraphs with Ricci curvature.
3) We provide a theoretically justified surrogate for Ricci curvature in the sparsification process to ensure its scalability.
4) Our proposed SEM with Ricci curvature-based sampling outperforms the existing state-of-the-art methods, especially in the challenging class-IL scenario.

II. Related Works

Our work is closely related to continual learning, continual graph learning, graph curvature, and graph sparsification.

A. *Continual Learning & Continual Graph Learning*

Machine learning models operating in the real world are expected to continually adapt to new tasks. However, they often encounter the catastrophic forgetting problem, *i.e.*, an abrupt performance decrease on previous tasks after learning new tasks. To resolve this challenge, various approaches have been developed and they can be roughly divided into three categories, *i.e.*, regularization based methods, parametric isolation based methods, and memory replay based methods. Regularization based methods prevent drastic modification to model parameters that are important to previous tasks through different constraints [24]–[27]. Parametric isolation based methods protect the parameters that are important to the previous tasks by adaptively allocating new parameters for new tasks [28]–[32]. Finally, memory replay based methods alleviate the forgetting problem by replaying representative data examples stored from previous tasks on the model when learning new tasks [6]–[8], [33]–[35].

Recently, continual learning on graph data is also attracting increasingly more attention due to its enormous value in various practical scenarios. For example, a community detection model has to keep adapting to nodes from newly emerged communities in social networks, a document classifier needs to keep learning to distinguish documents of newly emerged research areas in citation networks, *etc*. To this end, several methods have been introduced for continual graph learning [9], [23], [36]–[49]. These methods include regularization based ones like topology-aware weight preserving (TWP) [38] which preserves crucial parameters and topological structures of the previous tasks via regularization terms, parametric isolation based approaches like HPNs [37] which adaptively select different parameter combinations for different tasks, and memory replay based methods like ER-GNN [9] which stores representative nodes from the previous tasks in a buffer which are replayed when learning new tasks, and SSM [23] that stores computation subgraphs sparsified via random sampling. Our work is also based on memory replay. The key advantage of our model is that we can explicitly store the most informative topological information with a manageable space complexity, which exhibits superior performance in the experiments.

Finally, it is also worth noting the essential difference between continual graph representation learning, dynamic graph representation learning [43], [50]–[55], and few-shot graph learning [56]–[58]. Dynamic graph representation learning focuses on capturing the temporal dynamics of nodes with all previous information being accessible. On the contrary, continual graph representation learning focuses on alleviating the forgetting of previous tasks, therefore the previous data is inaccessible when learning new tasks. Few-shot graph learning aims at fast adapting the model to new tasks, which adopts a completely different setting. During training, few-shot learning models have access to data of all tasks simultaneously, while models under the continual learning setting can only access the data of the current task and are not allowed to access previous data. During testing, few-shot learning models are evaluated on new tasks and need to be fine-tuned with the test data, while the continual learning models are evaluated over existing tasks without any new data for fine-tuning.

B. *Graphs Curvatures & Graph Sparsification*

The vast majority of GNNs adopt the message passing paradigm [1], [2], [59]–[64], *i.e.* iteratively propagating the information of each node to its neighbors along the edges. Therefore, the information flow is largely determined by the topological structures of the graph. To investigate how smoothly the information travels on graphs and detect the information bottlenecks, the Ricci curvature, which is calculated based on the topological structures, is adopted as a powerful tool [11], [12]. The concept of curvature originates in the research on differential geometry for studying manifolds. As its discrete version, graph curvature can be intuitively understood as a metric on the graph bottleneck. For example, given the nodes u and v in Figure 1 (b), with a positive curvature in a complete graph, there are multiple paths connecting them and the information can easily flow between u and v. While in the discrete hyperbolic scenario with negative curvature, there is no other paths for the information to flow between u and v except the edge e_{uv}. The Ricci curvature has different formulations. The Forman-Ricci curvature [65], [66] is defined based on the combinatorial properties of the graph with a relatively low computation complexity. But its definition is biased to the negative curvatures. The Ollivier-Ricci curvature [67] on two nodes is defined based on the difference between their Wasserstein distance and shortest path distance. Its local quantities are hard to control and it has a higher computation complexity. Targeting these drawbacks, the Balanced Forman curvature [11] is proposed and is theoretically proven to be negatively correlated to the information bottlenecks when applying MPNNs on graph data. Therefore, we also adopt the Balanced Forman curvature to detect the most informative edges for sparsifying the computation subgraphs.

Graph sparsification has been extensively studied in the past few years [13]–[21]. However, unlike the curvature based sparsification proposed in our work, these methods are not specially developed to preserve the most informative neighbors for training GNNs. For instance, several prominent works have been presented to preserve certain predefined metrics or statistics on graphs [13]–[18]. In addition, these approaches may also encounter high computational burdens. Recently, various GNN explanation [19], [20], [68]–[71] or graph denoising [21] techniques have been developed to find the most informative structures. These methods, however, typically require training another network or iterative optimizations to explain one trained GNN, which will result in more resource consumption and are not suitable for continual learning. Despite this, some neighborhood sampling strategies could be adopted for our proposed SEM. For example, GraphSAGE [3] utilizes randomly sampled neighbors at each layer to reduce the computational burden. Similarly, DropEdge [72] randomly removes edges as a regularization. In experiments, besides curvature based sparsification, we will also adopt two random sampling based sparsification strategies [23] as baselines. One is uniform sampling, and the other is an importance sampling that samples the neighbors iteratively based on the node degree distribution.

III. Methods

In this section, we first introduce the preliminaries that include the basic concepts, notations, and learning settings. Next, we explain how memory replay works in traditional continual learning with independent data examples and why applying memory replay with GNNs on individual graph nodes can result in severe information loss. After that, we explain the memory explosion problem triggered by directly storing the complete topological information. Finally, we introduce our proposed solution for topology sparsification.

A. Preliminaries

Graph representation learning research focuses on node- or graph-level representations. Graph-level representation learning studies independent graph examples without connections among individual examples. It differs from node-level representation learning, in which the rich topological connections among individual examples have to be carefully considered. In this work, we focus on the node-level continual graph representation learning, which aims to continuously accommodate the new types (classes) of emerging nodes (new tasks) and their associated edges without interfering with the performance over existing nodes (previous tasks). With this setting, a model is trained on a sequence of tasks (subgraphs): $\mathcal{S} = \{\mathcal{G}_1, \mathcal{G}_2, ..., \mathcal{G}_T\}$. Each $\mathcal{G}_\tau$ contains nodes belonging to a unique set of classes in its node set $\mathbb{V}_\tau$. The associated edge set is $\mathbb{E}_\tau$, in which an edge e_{uv} denotes the existence of an edge connecting node u and v. $\mathbb{E}_\tau$ is often represented as the adjacency matrix $\mathbf{A}_\tau \in \mathbb{R}^{|\mathbb{V}_\tau|\times|\mathbb{V}_\tau|}$, where every non-zero entry corresponds to an edge in $\mathbb{E}_\tau$. $\mathbf{A}_\tau$ can be normalized as $\hat{\mathbf{A}}_\tau = \mathbf{D}_\tau^{-\frac{1}{2}}\mathbf{A}_\tau\mathbf{D}_\tau^{-\frac{1}{2}}$, where $\mathbf{D}_\tau \in \mathbb{R}^{|\mathbb{V}_\tau|\times|\mathbb{V}_\tau|}$ is the degree matrix that contains the degree of each node (number of connected edges) in its diagonal entries. Each node $u \in \mathbb{V}_\tau$ has a feature vector $\mathbf{x}_u \in \mathbb{R}^d$, and a label $\mathbf{y}_u \in \{0,1\}^C$, where C is the number of all possible classes. GNNs generate the representation for a node u based on a computation subgraph denoted as $\mathcal{G}_{\tau,u}^{sub}$, which is a subgraph of $\mathcal{G}_\tau$. In the following, $\mathcal{G}_u^{sub}$ without the graph index will be used for simplicity. Finally, we denote the L-hop neighbors of u as $\mathcal{N}^L(u)$ containing the nodes within a distance of L from u, *i.e.*, for a node $u \in \mathbb{V}_i$:

$$\mathcal{N}^L(u) = \{v \in \mathbb{V}_\tau | \mathrm{d}(u,v) = L\}, \tag{1}$$

where $\mathrm{d}(\cdot,\cdot)$ denotes the shortest path distance between two nodes. Specially, we have $\mathcal{N}^0(u) = \{u\}$. The vast majority of GNNs follow the message passing neural network (MPNN) paradigm [2]. In this work, we focus on the MPNNs and refer all GNNs to MPNNs in the following.

B. Memory Replay on Graphs

In this subsection, we first introduce how memory replay works in traditional continual learning, and then derive the information loss of directly applying memory replay to store individual graph nodes. Finally, we introduce the challenge of storing the topological information of graph data.

Traditional continual learning can be described as training a model $\mathrm{f}(\cdot;\boldsymbol{\theta})$ on a task sequence of length T with the accompanied datasets $\mathbb{D}_\tau = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^{n_\tau}\}$ ($\tau \in \{1, ..., T\}$). The dataset for τ-th task $\mathbb{D}_\tau$ is only available when learning this task, and becomes inaccessible afterward. To alleviate the forgetting problem, memory replay based methods typically

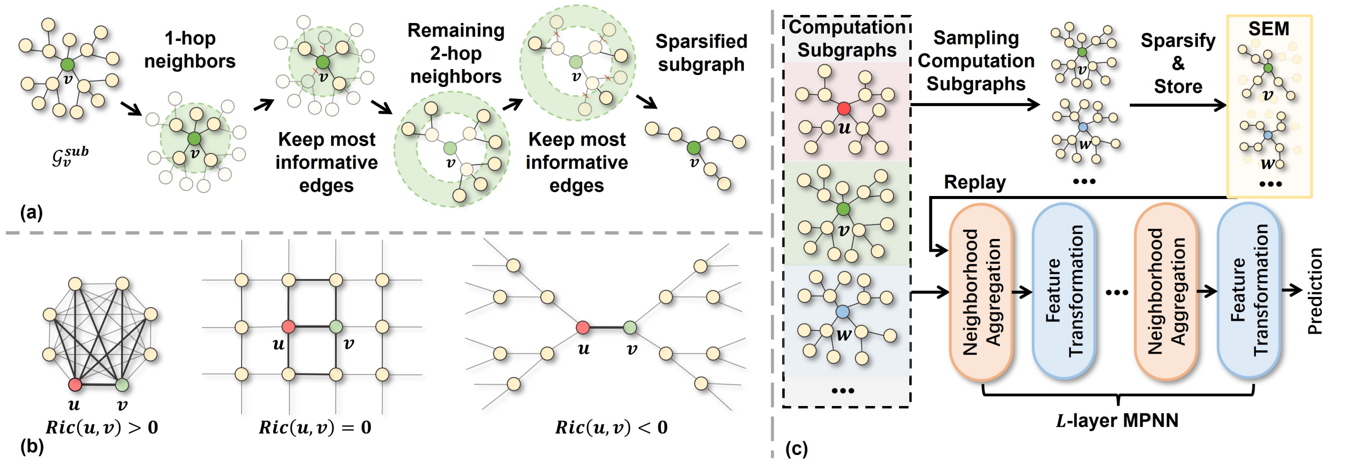


Fig. 1. (a) The iterative sparsification process for a computation subgraph containing 2-hop neighbors. (b) Illustration of edges with different curvatures. (c) The pipeline of SEM.

maintain a memory buffer $\mathcal{B}$ containing the representative data from the previous tasks, which are replayed to the model when learning new tasks. A straightforward way to utilize $\mathcal{B}$ is through an auxiliary loss:

$$\mathcal{L} = \underbrace{\sum_{\mathbf{x}_i \in \mathbb{D}_\tau} l(f(\mathbf{x}_i;\boldsymbol{\theta}), \mathbf{y}_i)}_{\text{loss of the current task } \mathcal{L}_\tau} + \lambda \underbrace{\sum_{\mathbf{x}_j \in \mathcal{B}} l(f(\mathbf{x}_j;\boldsymbol{\theta}), \mathbf{y}_j)}_{\text{auxiliary loss } \mathcal{L}_{aux}}, \quad (2)$$

where the loss function is denoted as $l(\cdot,\cdot)$, and the contribution of auxiliary loss is balanced by λ. Besides directly optimizing an auxiliary loss, there are also other ways to prevent forgetting with the stored data in $\mathcal{B}$. For example, GEM [7] calibrates the gradients of $\mathcal{L}_\tau$ with the gradients of $\mathcal{L}_{aux}$ to prevent increasing the loss on previous tasks ($\mathcal{L}_{aux}$); iCaRL [6] directly uses the representations of the stored data ($\{f(\mathbf{x}_i;\boldsymbol{\theta}), \mathbf{x}_i \in \mathcal{B}\}$) as prototypes to classify new data. For different approaches, we always have to regenerate their representations based on the buffered data. Traditional continual learning deals with independent data samples, and regenerating the representation $f(\mathbf{x}_i;\boldsymbol{\theta})$ only takes $\mathbf{x}_i$ itself as input. Therefore, the memory space complexity for replaying the representation of one independent example (node) is $\mathcal{O}(1)$.

To capture the rich topological information within a graph, the representation of a node not only depends on itself but also depends on its neighboring nodes and the accompanied edges. This will significantly increase the memory space complexity for replaying a graph node. In the following, we take the Message Passing Neural Networks (MPNNs) as an example. The rule for updating the hidden representation of a node u at layer $l+1$ is:

$$\mathbf{m}_u^{l+1} = \sum_{v \in \mathcal{N}^1(u)} \mathrm{M}_l(\mathbf{h}_u^l, \mathbf{h}_v^l, \mathbf{x}_{u,v}^e; \boldsymbol{\theta}_l^{\mathrm{M}}), \quad (3)$$

$$\mathbf{h}_u^{l+1} = \mathrm{U}_l(\mathbf{h}_u^l, \mathbf{m}_u^{l+1}; \boldsymbol{\theta}_l^{\mathrm{U}}), \quad (4)$$

where $\mathbf{h}_u^l$, $\mathbf{h}_v^l$ denote the hidden representations at layer l, $\mathbf{x}_{uv}^e$ denotes the possible edge features, the function $\mathrm{M}_l(\cdot,\cdot,\cdot;\boldsymbol{\theta}_l^{\mathrm{M}})$ generates message $\mathbf{m}_u^{l+1}$ from neighboring nodes, and the function $\mathrm{U}_l(\cdot,\cdot;\boldsymbol{\theta}_l^{\mathrm{U}})$ aggregates and updates the neighborhood information $\mathbf{m}_u^{l+1}$ into the representation $\mathbf{h}_u^l$ of the target node u. Given a L-layer MPNN, we can simplify the representation of a node u as:

$$\mathbf{h}_u^L = \mathrm{MPNN}(\mathcal{G}_u^{sub}; \boldsymbol{\Theta}), \quad (5)$$

where $\mathrm{MPNN}(\cdot;\boldsymbol{\Theta})$ denotes the composition of the message functions and the update functions at all layers. According to the updating rule, in L-layer MPNNs, $\mathcal{G}_u^{sub}$ would contain the L-hop neighbors $\mathcal{N}^L(v)$ and the accompanied edges.

To perform memory replay over graphs, the naive approach which stores individual nodes will lose the important explicit topological information of the computation subgraph $\mathcal{G}_u^{sub}$. A proper way should be to store representative computation subgraphs of existing tasks. However, due to the rich connections among different nodes, the size of different computation subgraphs varies and could be extremely large. Supposing the average node degree of $\mathcal{G}_v^{sub}$ is d, then the expected space complexity of storing its nodes would be $\mathcal{O}(d^L)$ (number of edges not counted yet), which can easily exceed the memory buffer size. For example, the average degree of the Reddit-CL dataset is 492, and the maximal degree is 21,657, which would easily result in intractable memory consumption.

C. Graph Sparsification via Information Bottlenecks

Intuitively, the nodes and edges in $\mathcal{G}_u^{sub}$ do not contribute equally to the representation of the target node u and we may need to keep the most prominent ones for memory replay. Since GNNs propagate information via edges, different topological connection patterns between two nodes largely determine the amount of information from one node to the other. Specifically, the neighbors with multiple paths to u would contribute more than those with few paths, as shown in Figure 1. Mathematically, this *hardness* for information propagation can be described with graph curvature. In this work, we adopt the Balanced Forman curvature [11] (one specific type of Ricci curvature) which is theoretically justified to be negatively correlated with the information bottleneck. Formally, the curvature is defined as:

Definition 1 (Ricci curvature)**.** *Given an edge e_{uv}, its curvature is defined as:*

$$Ric(u,v) := \frac{1}{d_u} + \frac{1}{d_v} - 2 + 2\frac{|\#_{\triangle}(u,v)|}{\max\{d_u,d_v\}} + \frac{|\#_{\triangle}(u,v)|}{\min\{d_u,d_v\}} + \frac{\gamma_{\max}^{-1}}{\max\{d_u,d_v\}}(|\#_{\square}^{u}(u,v)| + |\#_{\square}^{v}(u,v)|), \tag{6}$$

where d_u denotes the degree of a node, $\#_{\triangle}(u,v)$ and $\#_{\square}^{v}(u,v)$ are the higher order paths which will be detailed in the following, and $\gamma_{\max}^{-1}$ normalizes the number of 4-cycles according to how many cycles have shared edges.

1) *$\#_{\triangle}(u,v) := \mathcal{N}^1(u) \cap \mathcal{N}^1(v)$, i.e., the number of triangles formed with u, v and one of their common 1-hop neighbors.*
2) *$\#_{\square}^{u}(u,v) := \{w \in \mathcal{N}^1(u) | w \neq v, \exists q \in (\mathcal{N}^1(w) \cap \mathcal{N}^1(v)) \backslash \mathcal{N}^1(u)\}$. $\#_{\square}^{u}(u,v)$ counts the number of 4-cycles based at edge $e_{u,v}$, i.e., the path of length 3 propagating information from u to v. The second predicate excludes the paths of length 2, which are counted in the triangles $\#_{\triangle}(u,v)$.*
3) *$\gamma_{\max} = \max\big\{ \max_{p \in \#_{\square}^{u}(u,v)}\{|\mathcal{N}^1(p) \cap \#_{\square}^{v}(u,v) \backslash \mathcal{N}^1(u)| - 1\}, \max_{q \in \#_{\square}^{v}(u,v)}\{|\mathcal{N}^1(q) \cap \#_{\square}^{u}(u,v) \backslash \mathcal{N}^1(v)| - 1\}\big\}$ denotes the maximal number of 4-cycles based at edge $e_{u,v}$ traversing a common node.*

Note that although the curvature is defined based on each edge e_{uv}, it does not only count the contribution of e_{uv}. As shown in Figure 1(b), $\mathrm{Ric}(u,v)$ describes how easily information can be propagated from u to v based on topological connections of different lengths of paths around e_{uv}. Based on the Ricci curvature, a computation subgraph $\mathcal{G}_u^{sub}$ can be sparsified by selecting the most informative structures for representing the target node u. Specifically, to ensure the connectivity of the sparsified graph, we start by first sampling the 1-hop neighbors $\mathcal{N}^1(u)$ according to the curvature of the edges connecting u and nodes in $\mathcal{N}^1(u)$. Then, we sample the 2-hop neighbors that are connected to the selected 1-hop neighbors according to the curvature of the edges connecting them. By repeating this process, we sample a fixed size subset of nodes from 1-hop to L-hop neighbors. Together with the associated edges, we obtain a sparsified computation subgraph $\bar{\mathcal{G}}_u^{sub}$. However, the above process requires computing the curvature of a large number of edges in $\mathcal{G}_u^{sub}$. For a graph with $|\mathbb{E}|$ edges and maximal node degree as $d_{\max}$, the computation complexity of the curvature calculation for all edges is $|\mathbb{E}|d_{\max}^2$. On dense graphs with large $|\mathbb{E}|$ and $d_{\max}$, this would be intractable.

Fortunately, to sample the nodes, the exact values of the curvature are not essential. Instead, only the relative magnitudes of the curvatures matter, and thus we may derive surrogate metrics whose values are positively correlated to the curvature. Considering that the curvature describes the easiness of information flow on graphs, which is closely related to the graph diffusion process, the following theorem is derived:

Theorem 1. *Given a graph $\mathcal{G}_v^{sub}$ and the accompanied node set $\mathbb{V}_v$. For a graph (probability) diffusion process starting at v, we denote the probability distribution over a specific node u after i steps as $p^i(v,u)$. Accordingly, at the beginning, $p^0(v,v) = 1$ and $p^0(v,w) = 0$ for $w \in \mathbb{V}_v \backslash \{v\}$. Setting the rule of propagating the probability mass from node v to neighbors as: $\frac{1}{d_v}$ stays at v, and $\frac{1}{d_v}$ is propagated to each neighbor, the probability distribution on each neighbor $u \in \mathcal{N}^1(v)$ after a 3-step diffusion, i.e. $p^3(v,u)$, is positively correlated to* $\mathrm{Ric}(u,v)$.

Algorithm 1 Computation subgraph sparsification

```
1:  Input: $\mathcal{G}_u^{sub}$, node set $\mathbb{V}^{sub}$, edge set $\mathbb{E}^{sub}$, memory budget
    $\{k_l | l = 1, ..., L\}$.
2:  Output: Sparsified computation subgraph $\bar{\mathcal{G}}_u^{sub}$
3:  Initialize a node set $\mathbb{V} = \{u\}$, an empty edge set $\mathbb{E}$.
4:  for each $l \leftarrow 1$ to $L$ do
5:      Initialize a temporary node set $\mathbb{V}_{temp} = \{u\}$
6:      Initialize a probability set $\mathbb{P} = \{\}$
7:      for each $v \in \mathbb{V}_{temp}$ do
8:          for each $w \in \mathcal{N}^1(v)$ do
9:              Compute $p^3(w,v)$
10:             $\mathbb{P} = \mathbb{P} \cup \{p^3(w,v)\}$
11:         end for each
12:     end for each
13:     Sample a set of $k_l$ nodes $\mathbb{V}_{samp}$ according to $\mathbb{P}$
14:     $\mathbb{V} = \mathbb{V} \cup \mathbb{V}_{samp}$
15:     for each $v \in \mathbb{V}_{temp}$ do
16:         for each $w \in \mathbb{V}_{samp}$ do
17:             if $e_{w,v} \in \mathbb{E}^{sub}$ then
18:                 $\mathbb{E} = \mathbb{E} \cup \{e_{w,v}\}$  ▷ Store the accompanied
    edges
19:             end if
20:         end for each
21:     end for each
22:     $\mathbb{V}_{temp} = \mathbb{V}_{samp}$
23: end for each
24: Construct $\bar{\mathcal{G}}_u^{sub}$ with $\mathbb{V}$ and $\mathbb{E}$.
```

According to Theorem 1, a higher $p^3(v,u)$ indicates a higher curvature between v and u, *i.e.*, node v has a higher contribution to the representation of u in the message passing. Therefore, the diffusion probability distribution can be utilized as a surrogate for the Ricci curvature when sampling the most informative structures. Specifically, given a computation subgraph $\mathcal{G}_u^{sub}$ containing the neighbors from 1-hop to L-hop, we set a fixed memory budget K as the total number of nodes to sample from each hop. Denoting the budget for the l-th hop as k_l, we have $\sum_{l=1}^{L} k_l = K$. Then the detailed sampling procedure is described in Algorithm 1. The probability computation is shown in the form of for-loop for clarity. In practice, the random walk visiting probability of multiple nodes is implemented in parallel with mature deep learning libraries like Deep Graph Library (DGL).

Since the memory budget K is constant regardless of the graphs or models, the memory space complexity for replaying one node is reduced to $\mathcal{O}(1)$, which is manageable and similar to the traditional continual learning setting. With the sparisified computation subgraphs stored in the Subgraph Episodic

TABLE I
STATISTICS OF DATASETS AND TASK SPLITTINGS

Dataset	CoraFull-CL [74]	Arxiv-CL [1]	Reddit-CL [3]	Products-CL [2]
# nodes	19,793	169,343	232,965	2,449,029
# edges	130,622	1,166,243	114,615,892	61,859,140
# classes	70	40	40	47
# tasks	35	20	20	23

Memory $\mathcal{SEM}$, the loss of learning on task τ becomes:

$$\mathcal{L} = \underbrace{\sum_{u \in \mathbb{V}_\tau} l(\text{MPNN}(\mathcal{G}_u^{sub}; \boldsymbol{\Theta}), \mathbf{y}_u)}_{\text{loss of the current task } \mathcal{L}_\tau} + \lambda \underbrace{\sum_{\bar{\mathcal{G}}_v^{sub} \in \mathcal{SEM}} l(\text{MPNN}(\bar{\mathcal{G}}_v^{sub}; \boldsymbol{\Theta}), \mathbf{y}_v)}_{\text{auxiliary loss } \mathcal{L}_{aux}} . \quad (7)$$

Unlike in traditional learning which selects λ manually, to overcome the severe class imbalance problem in graph datasets, we choose to balance the loss with the class sizes as shown in Section IV-B3.

IV. Experiments

In this section, we aim to answer the following three research questions: **RQ1**: Whether storing subgraphs (sparsified) guarantees a better performance compared to only storing nodes? **RQ2**: How does the performance change with different sparsification rates (different amounts of nodes and edges are removed)? **RQ3**: Whether our proposed model can outperform existing state-of-the-art methods in both class-IL and task-IL scenarios? Our codes will be available online upon acceptance.

A. Datasets

We adopted four large public datasets following the settings of Continual Graph Learning Benchmark (CGLB) [73], including the datasets with up to millions of nodes, hundreds of millions of edges, and tens of tasks, which are very challenging for both class-IL and task-IL scenarios. Among the datasets, CoraFull-CL [74] and Arxiv-CL [75] are citation networks, Reddit-CL is a graph constructed from Reddit posts, and Products-CL [75] is an Amazon product co-purchasing network. For all datasets, each task contains two classes. For each class, 60% of the data are used for training, 20% are used for validation, and the remaining 20% are used for testing. Originally, these datasets were not for continual learning, and were proposed by different works. The links to the original sources, as well as detailed dataset statistics are included in Table I.

B. Experimental Settings

1) Continual learning setting and model evaluation.: In continual learning, a model continuously learns a sequence of tasks. During training, the model has access only to the data of the current task, while during testing, the model is expected to perform well on all previously learned tasks. The setting is further divided into class-incremental (class-IL) and task-incremental (task-IL) scenarios according to whether the task indicators are given.

Suppose the model learns on a citation network with a two-task sequence $\{$(*physics*, *chemistry*), (*biology*, *math*)$\}$. In class-IL scenario, after training, the model is required to classify a given document into one of the four classes. In task-IL scenario, the model is only required to classify a document into to (*physics*, *chemistry*) or (*biology*, *math*), while cannot distinguish between *physics* and *biology* or *chemistry* and *math*. Generally speaking, given a task sequence containing T tasks and each task contains n classes, during testing, task-IL only requires a model to pick a class for a given node from n classes, while class-IL requires a model to pick from $T \times n$ classes. In our experiments, the datasets contain up to 35 tasks and 70 classes. In the case with 35 tasks and each task contains 2 classes, after learning the entire task sequence, class-IL setting requires a model to pick the correct class from 70 classes, while task-IL still only requires the model to distinguish between 2 classes (task identity is given to indicate which two classes are being tested). Considering the difficulty mentioned above together with the forgetting issue, class-IL is more practical and challenging than task-IL. Besides task-IL and class-IL, domain-IL has also been studied and refers to a scenario in which the task is fixed but the domain of the data constantly changes. The difficult of domain-IL is typically between task-IL and class-IL. Existing works typically adopt task-IL or class-IL tasks, which can be naturally constructed from most real-world datasets. While the domain-IL scenario is usually studied for specific applications including knowledge graph [41] and recommender system [40], [76].

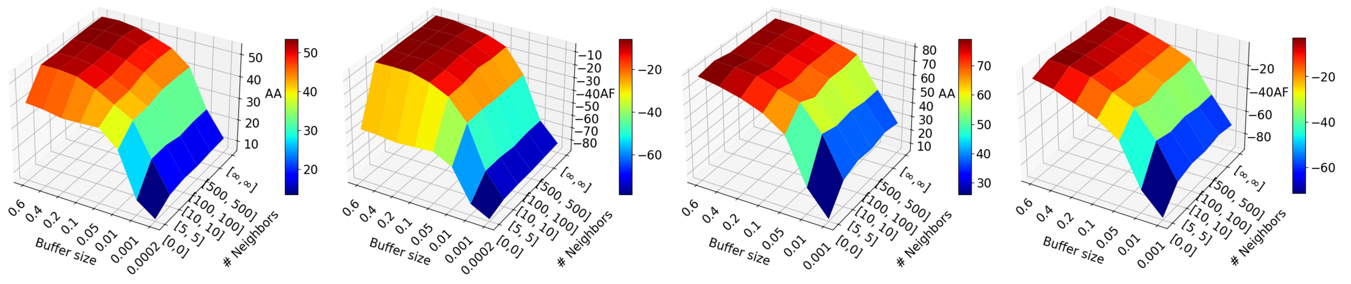
For continual learning models, the most thorough evaluation is the accuracy matrix $\mathrm{M}^{acc} \in \mathbb{R}^{T \times T}$. Each entry $\mathrm{M}^{acc}_{i,j}$ denotes the model's accuracy on task j after learning task i. Then each row $\mathrm{M}^{acc}_{i,:}$ shows the model's accuracy on all previous tasks after learning task i, and each column $\mathrm{M}^{acc}_{:,j}$ shows how the model's accuracy on task j changes when being trained consecutively on all T tasks. To derive a single numeric value for evaluation, the average accuracy (AA) $\frac{\sum_{i=1}^{T} \mathrm{M}^{acc}_{T,i}}{T}$ and average forgetting (AF) $\frac{\sum_{i=1}^{T-1} \mathrm{M}^{acc}_{T,i} - \mathrm{M}^{acc}_{i,i}}{T-1}$ after learning all T tasks will be used. All experiments are repeated 5 times on one Nvidia Titan Xp GPU, and results are reported with average performance and standard deviations.

2) Baselines and model settings: Our adopted baselines include the methods specially designed for continual graph learning: Experience Replay based GNN (ERGNN) [9] and Topology-aware Weight Preserving (TWP) [38], and milestone works designed for traditional continual learning on Euclidean data but also applicable to GNNs: Elastic Weight Consolidation (EWC) [25], Learning without Forgetting (LwF) [26], Gradient Episodic Memory (GEM) [7], and Memory Aware Synapses (MAS) [27]). We also adopt joint training as the upper bound. A jointly trained model is simultaneously trained on all tasks. Therefore, it has no forgetting problem and can serve as an upper bound on the continual learning

[1]https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv
[2]https://ogb.stanford.edu/docs/nodeprop/#ogbn-products

TABLE II
PERFORMANCE COMPARISONS UNDER CLASS-IL ON 4 DATASETS WITH SGC BACKBONE (↑ HIGHER MEANS BETTER). THE BEST AND THE SECOND BEST PERFORMANCE OBTAINED BY CONTINUAL LEARNING MODELS ARE HIGHLIGHTED BY BOLDFACE AND UNDERLINE, RESPECTIVELY.

Continual learning techniques	CoraFull		OGB-Arxiv		Reddit		OGB-Products	
	AA/% ↑	AF/% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑
Fine-tune	3.5±0.5	-95.2±0.5	4.9±0.0	-89.7±0.4	5.9±1.2	-97.9±3.3	7.6±0.7	-88.7±0.8
EWC [25]	52.6±8.2	-38.5±12.1	8.5±1.0	-69.5±8.0	10.3±11.6	-33.2±26.1	23.8±3.8	-21.7±7.5
MAS [27]	6.5±1.5	-92.3±1.5	4.8±0.4	-72.2±4.1	9.2±14.5	-23.1±28.2	16.7±4.8	-57.0±31.9
GEM [7]	8.4±1.1	-88.4±1.4	4.9±0.0	-89.8±0.3	11.5±5.5	-92.4±5.9	4.5±1.3	-94.7±0.4
TWP [38]	62.6±2.2	-30.6±4.3	6.7±1.5	-50.6±13.2	8.0±5.2	-18.8±9.0	14.1±4.0	-11.4±2.0
LwF [26]	33.4±1.6	-59.6±2.2	9.9±12.1	-43.6±11.9	86.6±1.1	-9.2±1.1	48.2±1.6	-18.6±1.6
ER-GNN [9]	34.5±4.4	-61.6±4.3	21.5±5.4	-70.0±5.5	82.7±0.4	-17.3±0.4	48.3±1.2	-45.7±1.3
SSM-uniform [23]	73.0±0.3	-14.8±0.5	47.1±0.5	-11.7±1.5	94.3±0.1	-1.4±0.1	62.0±1.6	-9.9±1.3
SSM-degree [23]	75.4±0.1	-9.7±0.0	48.3±0.5	-10.7±0.3	94.4±0.0	-1.3±0.0	63.3±0.1	-9.6±0.3
Joint (Not under continual setting)	81.2±0.4	-	51.3±0.5	-	97.1±0.1	-	71.5±0.1	-
SEM-curvature (Ours)	**77.7±0.8**	-10.0±1.2	**49.9±0.6**	-8.4±1.3	**96.3±0.1**	-0.6±0.1	**65.1±1.0**	-9.5±0.8



(a) OGB-Arxiv, AA (b) OGB-Arxiv, AF (c) CoraFull, AA (d) CoraFull, AF

Fig. 2. Influence of buffer size and sparsification levels on the performance of SEM on different datasets. (a) Average accuracy on Arxiv-CL dataset. (b) Average forgetting on Arxiv-CL dataset. (c) Average accuracy on CoraFull-CL dataset. (d) Average forgetting on CoraFull-CL dataset.

performance. Besides, fine-tune (without continual learning technique) is adopted as the lower bound. All models are implemented based on four popular backbone GNNs, *i.e.*, Graph Convolutional Networks (GCNs) [1], Simple Graph Convolution (SGC) [77], Graph Attention Networks (GATs) [78], and Graph Isomorphism Network (GIN) [79]. In the paper, we show the results on the SGC backbone, and the results of the other backbones are provided in Appendix A. For the backbone models, the input dimension is determined by the dimension of the node attribute vector in the given dataset, and the output dimension is determined by the number of classes in each task. To fairly compare different continual learning techniques, all backbones are set as 2-layer with 256 hidden dimensions. For the multi-head mechanism in GAT, we adopt 8 heads, each of which outputs 32 dimensions. Therefore the hidden dimension of GAT is also 256. For all backbone models, the hidden dimension is validated over 32, 128, 256, and 512 on the validation set, and 256 turns out to be the best option.

3) Class imbalance & class-IL classifier: According to Section IV-B1, the performance on different tasks contribute equally to the average accuracy. However, unlike the traditional continual learning with balanced datasets, the class imbalance problem is usually severe in graphs, of which the effect will be entangled with the effect of forgetting. To balance the data by simply choosing an equal number of graph nodes from each class is impractical. For example, in the Products-CL dataset, the largest class has 668,950 nodes, while the smallest contains only 1 node. Therefore, sampling an equal amount of nodes from each class would result in either deleting many classes without enough nodes or sampling a very small number of nodes from each class so that all classes can provide enough nodes. Moreover, deleting nodes in a graph would also change the original topological structures of the remaining nodes, which is undesired. To this end, we propose to rescale the loss according to the class sizes. Denoting the set of all classes as $\mathcal{C}$, and the number of examples of each class as $\{n_c \mid c \in \mathcal{C}\}$, we can calculate a scale for each class c to balance their contribution in the loss function as $s_c = \frac{n_c}{\sum_{i \in \mathcal{C}} n_i}$. Finally, the balanced loss is:

$$\begin{aligned}\mathcal{L} = &\sum_{v \in \mathbb{V}_\tau} l(f(\mathbf{e}_v;\boldsymbol{\theta}), y_v) \cdot s_{y_v} \\ &+ \sum_{\mathbf{e}_w \in \mathcal{SEM}} l(f(\mathbf{e}_w;\boldsymbol{\theta}), y_w) \cdot s_{y_w}, \end{aligned} \tag{8}$$

λ in Equation (7) is omitted as it will influence the balance of each class. Empirically, this choice results in significantly better performance for all methods.

The number of the output heads of a model in a standard classification task equals the number of classes and is fixed at the beginning. But in the class-IL scenario, the output heads continually increase along with the new classes. To better accommodate the new classes, cosine distance is adopted by a number of works [80]–[82] to modify the standard softmax classifier. In our experiments, all baselines except LwF adopt the standard classifier, since only LwF exhibits

TABLE III
PERFORMANCE COMPARISONS UNDER TASK-IL ON 4 DATASETS WITH SGC BACKBONE (↑ HIGHER MEANS BETTER). THE BEST AND THE SECOND BEST PERFORMANCE OBTAINED BY CONTINUAL LEARNING MODELS ARE HIGHLIGHTED BY BOLDFACE AND UNDERLINE, RESPECTIVELY.

Continual learning techniques	CoraFull		OGB-Arxiv		Reddit		OGB-Products	
	AA/% ↑	AF/%	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑	AA/% ↑	AF /% ↑
Fine-tune	56.0±4.2	-41.0±4.5	56.2±2.6	-36.2±2.6	79.5±24.2	-11.7±4.8	64.4±3.8	-31.1±4.4
EWC [25]	89.8±1.0	-5.1±0.5	71.5±0.6	-0.9±0.6	83.9±15.1	-2.0±1.5	87.0±1.4	-1.7±1.2
MAS [27]	92.2±0.9	-3.7±1.3	72.7±2.6	-18.5±2.5	61.1±7.1	-0.5±1.0	80.6±4.3	-13.7±3.7
GEM [7]	91.5±0.5	-1.9±0.9	81.1±1.7	-4.0±1.8	98.9±0.1	-0.5±0.1	87.7±1.8	-7.0±2.0
TWP [38]	94.3±0.9	-1.6±0.4	<u>89.4±0.4</u>	0.0±0.3	78.0±18.5	-0.2±0.4	81.8±3.3	-0.3±0.8
LwF [26]	93.8±0.1	-0.4±0.1	71.1±3.2	-1.5±0.8	98.6±0.1	-0.0±0.0	86.3±0.2	-0.5±0.1
ER-GNN [9]	86.3±1.0	-9.2±0.9	86.4±0.3	0.5±0.6	97.4±0.2	4.7±0.1	86.4±0.0	11.7±0.0
SSM-uniform [23]	95.3±0.5	0.2±0.5	88.5±0.6	-1.3±0.5	99.2±0.0	-0.2±0.0	93.1±0.8	-1.8±0.3
SSM-degree [23]	<u>95.8±0.3</u>	0.6±0.2	88.4±0.3	-1.1±0.1	<u>99.3±0.0</u>	-0.2±0.0	<u>93.2±0.7</u>	-1.9±0.0
Joint (Not under continual setting)	95.5±0.2	-	90.3±0.4	-	99.5±0.0	-	95.3±0.8	-
SEM-curvature (Ours)	**95.9±0.5**	0.7±0.4	**89.9±0.3**	-0.1±0.5	**99.3±0.0**	-0.2±0.0	**93.2±0.7**	-1.8±0.4

better performance through the classifier modified with the cosine distance.

C. Studies on Sparsification Levels and Buffer Sizes ($\mathrm{RQ1}, \mathrm{RQ2}$*)*

In Figure 2, we show the average accuracy and forgetting obtained with different buffer sizes and sparsification levels on OBG-Arxiv-CL and CoraFull-CL datasets. Buffer size (per class) denotes the number of nodes whose sparsified computation subgraphs are stored. For clarity, we covert the buffer size into the ratio of the dataset size. For thorough investigation, we vary the buffer size from 1 node per class (0.02% of the size of Arxiv-CL and 0.1% of CoraFull-CL) to the size of the entire training set (60%). The sparsification level is denoted as the number of neighbors to keep for each hop. Since the backbones are 2-layer, only the 2-hop neighbors are concerned.

To clearly denote the numbers of neighbors we store from neighbors at each hop, we enclose them in a square bracket, [n_1, n_2, ...], where n_i denotes the budget for the i-th hop. When the computation subgraph covers 2-hop neighborhood, there are two entries in the brackets, i.e. [n_1, n_2]. Accordingly, there are two extreme cases. First, [0, 0] denotes the situation when we only store the individual center nodes without considering the neighborhood (topology). Second, [∞, ∞] means that we have unlimited budget for each hop and store the entire computation subgraph. From Figure 2, we could see a significant increase in the performance when the buffer starts to store subgraphs ([5, 5]) compared to storing individual nodes ([0, 0]), while the performance gain brought by storing denser computation subgraphs (from [5, 5] to [∞, ∞]) is relatively low. This implies the importance of topological information and indicates the computation subgraphs are highly redundant and our sparsification strategies can effectively preserve the crucial information.

D. Comparisons for Class-IL Scenario ($\mathrm{RQ1}, \mathrm{RQ3}$*)*

In this subsection, we compare SEM and the baselines under the class-IL scenario. For Arxiv-CL, Products-CL, and Reddit-CL datasets, we choose the buffer size to be 400 per class. For CoraFull-CL, we only allow a budget of 60 per class. For the memory based baselines, we allow a budget of up to 800 per class for all datasets to highlight the advantage of SEM. As shown in Table II, under the class-IL setting, SEM-curvature not only outperforms the baselines with a large margin on all datasets, but also outperforms the two memory replay baselines based on uniform sampling and degree based sampling. The performance is even comparable to the joint training. To understand the learning dynamics, we visualize the accuracy matrices of the most representative methods including SEM, the two best baseline methods ER-GNN and LwF (based on knowledge distillation), and fine-tune. As shown in Figure 3, by comparing the columns of different matrices, we can find different forgetting patterns. First, SEM maintains a very stable performance of each task throughout the entire training process. The performance exhibits no abrupt decrease compared to the baselines. Second, the performance of ER-GNN on each task is not monotonically decreasing. Instead, the performance on a task may first decrease and experience a resurgence later. This may be the benefit of the memory buffer, which keeps driving the model to a favorable point for previous tasks. In contrast, the performance of LwF decreases monotonically. Fine-tuning a model in the class-IL scenario is ineffective as the knowledge of previous tasks is completely overwritten by new tasks.

E. Comparisons for Task-IL Scenario ($\mathrm{RQ1}, \mathrm{RQ3}$*)*

Finally, we also compare SEM with different baseline approaches in the task-IL scenario. As explained in Section IV-B1, during testing on each task, class-IL requires the model to pick a class from all learnt classes, while task-IL only require the model to distinguish between the classes within the given task. Therefore, class-IL is much more challenging especially when the number of classes are large. In our experiments, the datasets contain at least 40 classes (OGB-Arxiv and Reddit) and at most 70 classes (CoraFull), which significantly increase the learning difficulty of class-IL. Accordingly, as shown in Table III, task-IL is experimentally much less challenging than class-IL, and many more methods perform very well in this scenario.

However, our method still outperforms all baseline techniques on 4 different datasets. Moreover, on all datasets, our method obtains comparable performance with joint training (Joint). Considering that joint training is simultaneously

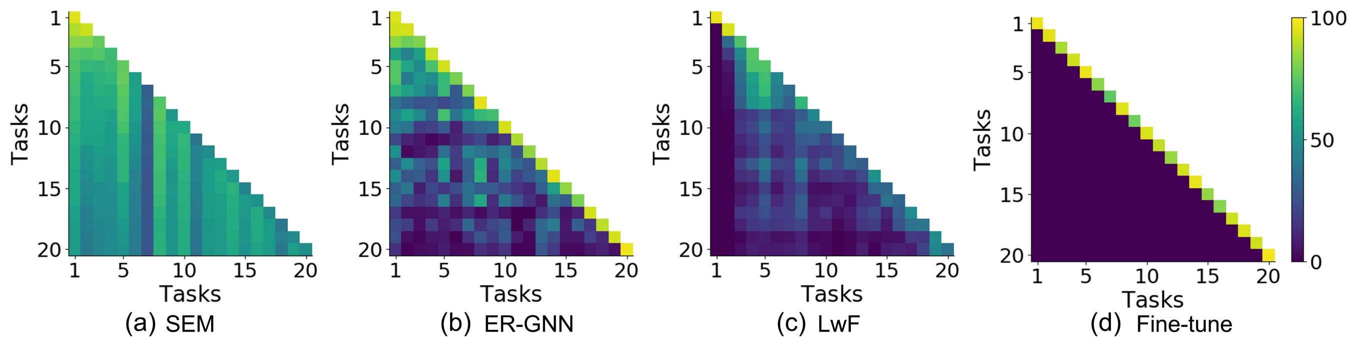


Fig. 3. From left to right: accuracy matrix of SEM, ER-GNN, LwF, and Fine-tune on Arxiv-CL dataset. In an accuracy matrix, an entry at the i-th row and j-th column is the model's accuracy on the j-th task after learning the i-th task.

TABLE IV
META-PATHS FROM v TO u VIA A RANDOM WALK OF 3 MOVES

Meta-path	1	2	3	4	5	6	7	8	9	10
Start	v	v	v	v	v	v	v	v	v	v
Intermediate node 1	v	v	u	u	v	u	p	p	p	p
Intermediate node 2	v	u	v	u	p	p	v	u	p	q
End	u	u	u	u	u	u	u	u	u	u

trained on all tasks with full topological information (without the forgetting issue), and is the upper bound on the continual learning models, the effectiveness of our method is very significant.

V. CONCLUSION

In this paper, we developed Ricci curvature based graph sparsification technique to perform continual graph representation learning. We employed Subgraph Episodic Memory (SEM) to store the topological information in the form of computation subgraphs. By leveraging Ricci curvature to assess the informativeness, we can reduce the memory consumption of a computation subgraph from $\mathcal{O}(d^L)$ to $\mathcal{O}(1)$, and enable GNNs to utilize the most informative topological information for memory replay. Our proposed method outperforms the existing state-of-the-art methods on 4 public datasets in both class-IL (more practical and challenging) and task-IL scenarios.

A. Proof Details

In this section, we theoretically analyze the correlation between the graph diffusion process and the curvature under the lazy random walk framework.

Proof. Since the diffusion process is modeled as a lazy random walk model in which the probability of staying at a node v is defined as $\frac{1}{d_v}$.To calculate the probability of starting at a v and arriving at another node u after 3 moves, we first divide the walks into 10 meta-paths in Table IV. Totally, we have 10 different paths. v and u are always the starting and ending nodes, and the choice of the two intermediate nodes could be different. In Table IV, p and q denote two nodes other than v and u. Since p and q do not refer to specific nodes, some seemingly different meta-paths actually refer to a same meta-path and are not listed, *e.g.* path $vpqu$ and $vqpu$, path $vppu$ and $vqqu$, *etc.*

Then we calculate the probability of walking from v to u via each meta-path. In our random walk process, the self-loop is added to each node and at each step the probability to not move is 0. In other words, a path vv does not imply that the walker stays at the original node, instead it chooses the self-loop of v. Since the probability of choosing each neighbor is equal, this setting ensures the probability of taking path vv to be $\frac{1}{d_v}$. Note that this does not include v into $\mathcal{N}^1(v)$, since $d_G(v,v)=0$, and $v\in\mathcal{N}^0(v)$. And we have $d_v-1=|\mathcal{N}^1(v)|$. Besides, we assume the graph to be a simple, *i.e.* two nodes can be connected via no more than one edge. The probabilities of each meta-path are:

$$p(vvvu)=\frac{1}{d_v}\cdot\frac{1}{d_v}\cdot\frac{1}{d_v}=\frac{1}{d_v^3} \tag{9}$$

$$p(vvuu)=\frac{1}{d_v}\cdot\frac{1}{d_v}\cdot\frac{1}{d_u}=\frac{1}{d_v^2\cdot d_u} \tag{10}$$

$$p(vuvu)=\frac{1}{d_v}\cdot\frac{1}{d_u}\cdot\frac{1}{d_v}=\frac{1}{d_v^2\cdot d_u} \tag{11}$$

$$p(vuuu)=\frac{1}{d_v}\cdot\frac{1}{d_u}\cdot\frac{1}{d_u}=\frac{1}{d_v\cdot d_u^2} \tag{12}$$

$$\begin{aligned}p(vvpu)&=\sum_{p\in\mathcal{N}^1(v)\cap\mathcal{N}^1(u)}\frac{1}{d_v}\cdot\frac{1}{d_v}\cdot\frac{1}{d_p}\\&=\sum_{p\in\mathcal{N}^1(v)\cap\mathcal{N}^1(u)}\frac{1}{d_v^2\cdot d_p}\end{aligned} \tag{13}$$

$$\begin{aligned}p(vupu)&=\sum_{p\in\mathcal{N}^1(u)}\frac{1}{d_v}\cdot\frac{1}{d_u}\cdot\frac{1}{d_p}\\&=\sum_{p\in\mathcal{N}^1(u)}\frac{1}{d_v\cdot d_u\cdot d_p}\end{aligned} \tag{14}$$

$$p(vpvu) = \sum_{p \in \mathcal{N}^1(v)} \frac{1}{d_v} \cdot \frac{1}{d_p} \cdot \frac{1}{d_v} = \sum_{p \in \mathcal{N}^1(v)} \frac{1}{d_v^2 \cdot d_p} \tag{15}$$

$$p(vpuu) = \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v} \cdot \frac{1}{d_p} \cdot \frac{1}{d_u} = \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v \cdot d_u \cdot d_p} \tag{16}$$

$$p(vppu) = \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v} \cdot \frac{1}{d_p} \cdot \frac{1}{d_p} = \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v \cdot d_p^2} \tag{17}$$

$$p(vpqu) = \sum_{p \in \mathcal{N}^1(v), q \in (\mathcal{N}^1(p) \cap \mathcal{N}^1(u) \backslash \{v\})} \frac{1}{d_v} \cdot \frac{1}{d_p} \cdot \frac{1}{d_q} \tag{18}$$

The Equation 9 to 12 correspond to the paths only concerned with Summing up the Equations 9 to 12, Equations 14 and 15 as p_1, we get:

$$p_1 = \frac{1}{d_v} \cdot \left(\frac{1}{d_v^2} + \frac{2}{d_v d_u} + \frac{1}{d_u^2} + \frac{|\mathcal{N}^1(u)|}{d_u d_p} + \frac{|\mathcal{N}^1(v)|}{d_v d_p}\right) \tag{19}$$

$$= \frac{1}{d_v} \cdot \left(\frac{1}{d_v^2} + \frac{1}{d_v d_u} + \frac{|\mathcal{N}^1(v)|}{d_v d_p}\right) + \frac{1}{d_u} \cdot \left(\frac{1}{d_v^2} + \frac{1}{d_v d_u} + \frac{|\mathcal{N}^1(v)|}{d_v d_p}\right) \tag{20}$$

Since both $(\frac{1}{d_v^2} + \frac{1}{d_v d_u} + \frac{|\mathcal{N}^1(v)|}{d_v d_p})$ and $(\frac{1}{d_v^2} + \frac{1}{d_v d_u} + \frac{|\mathcal{N}^1(v)|}{d_v d_p})$ are positive, the two terms of p_1 are positively correlated to the first two terms $\frac{1}{d_v}$ and $\frac{1}{d_u}$ of Equation 6, respectively. Whenever $\frac{1}{d_v}$ or $\frac{1}{d_u}$ increases, their corresponding part in p_1 would increase. Similarly, we will find the correspondence between the rest terms of Equation 6 and the random walk probabilities. Equations 13, 16, and 17 depends on the common neighbors of u and v, which corresponds to the triangle counter $\#_\triangle$ in Equation 6 By summing up these equations, we get:

$$p_2 = \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v^2 \cdot d_p} + \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v \cdot d_u \cdot d_p} \tag{21}$$

$$+ \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_v \cdot d_p^2} \tag{22}$$

$$= \sum_{p \in \mathcal{N}^1(v) \cap \mathcal{N}^1(u)} \frac{1}{d_p} \cdot \left(\frac{1}{d_v^2} + \frac{1}{d_v \cdot d_u} + \frac{1}{d_v \cdot d_p}\right) \tag{23}$$

By definition, $|\mathcal{N}^1(v) \cap \mathcal{N}^1(u)| = |\#_\triangle(u,v)|$. Since all summation terms in Equation 21 are positive, we can rewrite it in a more concise format by assuming all nodes in $\mathcal{N}^1(v) \cap \mathcal{N}^1(u)$ has a degree $d_p = \bar{d_p}$:

$$p_2 = |\#_\triangle(v,u)| \cdot \frac{1}{\bar{d_p}} \cdot \left(\frac{1}{d_v^2} + \frac{1}{d_v \cdot d_u} + \frac{1}{d_v \cdot \bar{d_p}}\right) \tag{24}$$

And the corresponding terms in Equation 6 is:

$$|\#_\triangle(u,v)| \cdot \left(\frac{2}{\max\{d_u, d_v\}} + \frac{1}{\min\{d_u, d_v\}}\right). \tag{25}$$

Since both expressions have positive coefficients on the term $|\#_\triangle(u,v)|$, they are also positively correlated. Finally, by definition, $|\#^u_\square(u,v)|$ or $|\#^v_\square(u,v)|$ refer to the number of 1-hop neighbors of u or v participating in a 4-cycle based at edge $e_{u,v}$. Therefore, the number of 4-cycles based at $e_{u,v}$ is $|\#_\square(u,v)| = \max(|\#^u_\square(u,v)|, |\#^v_\square(u,v)|)$. By observing Equation 18, by traversing p, q via $p \in \mathcal{N}^1(v), q \in (\mathcal{N}^1(p) \cap \mathcal{N}^1(u) \backslash \{v\})$, we are also counting the number of 4-cycles, **i.e.** $|\#_\square(u,v)|$. □

VI. Future Directions

Although our proposed method achieves impressive performance on multiple datasets under different learning scenarios, there are still several perspectives that can be considered to further improve the model.

First, our proposed method requires additional space for storing the sparsified computation subgraphs. Although we have demonstrated that the space consumption is acceptable on graphs up to millions of nodes and tens of millions of edges, some real-world graphs may be even larger and contain billions or even trillions of nodes. On such graphs, if the node attributes are highly diverse, maintaining a satisfying performance may require large memory buffers.

Second, with some databases, due to privacy reasons, storing information from the original graph may not be allowed. In this case, we may have to design additional modules to erase the identifying information before buffering the data.

Third, the idea of learning based graph augmentation from AutoGCL [83] could be borrowed into our sparsification process, and will be investigated in our future work. Specifically, we will focus on the following challenges.

- AutoGCL does not constrain the sizes of the augmented graphs, which is still subject to the memory explosion problem studied in our work. Therefore, how to constrain the size of the graphs generated by the learnable module is the topmost challenge.
- AutoGCL generate different views of a given graph. Each view contains both label relevant information and label irrelevant information that serves to decrease the mutual information between different views. In our sparsification process, the label irrelevant information is undesired. Therefore, how to retain only the label relevant information is the second challenge.

Fourth, contrastive learning has achieved significant successes recently. For instance, Hypergraph Contrastive Collaborative Filtering [84] boosts the representation quality for recommender system based on hypergraph structure encoding and contrastive learning. Directed Graph Contrastive Learning [85] proposes to generate different views by perturbing the Lapalacian matrix, so that the directed graph structure is not changed. The contrastive learning techniques is also promising for boosting the continual learning performance. Specifically, contrastive learning could avoid the dependency on the labels, which could significantly reduce the overfitting to each task and encourage the transferability of the learnt knowledge. In this way, the forgetting problem can also be alleviated. Towards this target, we will focus on two challenges.

- Although with less dependency on the task specific information, *i.e.* labels, contrastive learning based models still have to continually adapt to new distributions, and experience the forgetting problem. Moreover, in the class-IL setting, the final classifier, whose parameters are shared across different tasks, still needs supervised training and is subject to catastrophic forgetting.
- The existing continual learning works adopt task configurations constructed for supervised learning, and how to properly configure the tasks for the self-supervised learning setting requires careful consideration. For instance, how to split a given dataset into supervised and self-supervised parts, whether the model is allowed to access the self-supervised data all the time, etc.

Fifth, in this work, we target MPNNs working on homophilious graphs. However, learning graphs with heterophily are also practically important and are attracting increasingly more attention recently. For example, [86] systematically studies the challenges of learning on heterophilous graphs and proposes several techniques for designing effective heterophilous GNNs. [87] proposes a new constraint, *i.e.* homophily unnoticeability, to ensure that the Graph Injection Attack (GIA) preserves the graph homophily, so that the attack of GIA cannot be easily counteracted simply by homophily recovery. Due to its practical importance, extending our proposed SEM-curvature to heterophilous graphs also deserves investigation. To achieve this aim, we will focus on two challenges. First, SEM-curvature is designed for MPNNs based on the message passing strategy. Since simple message passing is not suitable for heterophilous graphs [86], the curvature based graph sparsification in SEM-curvature is not directly applicable to heterophilous graph learning. Therefore, how to adapt the framework for analyzing information propagation to heterophilous GNNs becomes a key challenge. Second, existing works on continual graph learning do not focus on heterophilous graphs, and the benchmark tasks constructed from heterophilous graphs are lacking.

VII. Acknowledgment

Dacheng Tao and Xikun Zhang are supported by Australian Research Council Projects FL-170100117 and IC-190100031. Dongjin Song is supported by NSF CNS-2154191, USDA NIFA 2022-67022-36603, and UConn Scholarship Facilitation Fund (SFF) Award.

References

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[2] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1263–1272.

[3] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, 2017, pp. 1024–1034.

[4] J. L. McClelland, B. L. McNaughton, and R. C. O'Reilly, "Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory." *Psychological Review*, vol. 102, no. 3, p. 419, 1995.

[5] D. Kumaran, D. Hassabis, and J. L. McClelland, "What learning systems do intelligent agents need? complementary learning systems theory updated," *Trends in Cognitive Sciences*, vol. 20, no. 7, pp. 512–534, 2016.

[6] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[7] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6467–6476.

[8] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 816–11 825.

[9] F. Zhou and C. Cao, "Overcoming catastrophic forgetting in graph neural networks with experience replay," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 4714–4722.

[10] C. Wang, Y. Qiu, and S. Scherer, "Bridging graph network to lifelong learning with feature interaction," 2020.

[11] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein, "Understanding over-squashing and bottlenecks on graphs via curvature," *arXiv preprint arXiv:2111.14522*, 2021.

[12] Z. Ye, K. S. Liu, T. Ma, J. Gao, and C. Chen, "Curvature graph network," in *ICLR*, 2019.

[13] C. Hübler, H.-P. Kriegel, K. Borgwardt, and Z. Ghahramani, "Metropolis algorithms for representative subgraph sampling," in *ICDM*. IEEE, 2008, pp. 283–292.

[14] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen, "Sparsification of influence networks," in *KDD*, 2011, pp. 529–537.

[15] D. Calandriello, A. Lazaric, I. Koutis, and M. Valko, "Improved large-scale graph learning through ridge spectral sparsification," in *ICML*. PMLR, 2018, pp. 688–697.

[16] A. Chakeri, H. Farhidzadeh, and L. O. Hall, "Spectral sparsification in spectral clustering," in *ICPR*. IEEE, 2016, pp. 2301–2306.

[17] J. Leskovec and C. Faloutsos, "Sampling from large graphs," in *KDD*, 2006, pp. 631–636.

[18] A. S. Maiya and T. Y. Berger-Wolf, "Sampling community structure," in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 701–710.

[19] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *NeurIPS*, vol. 32, 2019.

[20] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *NeurIPS*, vol. 33, pp. 19 620–19 631, 2020.

[21] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *ICML*. PMLR, 2020, pp. 11 458–11 468.

[22] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Hierarchical representation learning in graph neural networks with node decimation pooling," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[23] X. Zhang, D. Song, and D. Tao, "Sparsified subgraph memory for continual graph representation learninggated information bottleneck for generalization in sequential environments," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022.

[24] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 3762–3773.

[25] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[26] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[27] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 139–154.

[28] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi, "Supermasks in superposition," *arXiv preprint arXiv:2006.14769*, 2020.

[29] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, and Y. Fu, "Large scale incremental learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 374–382.

[30] J. Yoon, S. Kim, E. Yang, and S. J. Hwang, "Scalable and order-robust continual learning with additive parameter decomposition," in *International Conference on Learning Representation*, 2020.

[31] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[32] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[33] L. Caccia, E. Belilovsky, M. Caccia, and J. Pineau, "Online learned continual compression with adaptive quantization modules," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1240–1250.

[34] A. Chrysakis and M.-F. Moens, "Online continual learning from imbalanced data," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1952–1961.

[35] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[36] J. Cai, X. Wang, C. Guan, Y. Tang, J. Xu, B. Zhong, and W. Zhu, "Multimodal continual graph learning with neural architecture search," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1292–1300.

[37] X. Zhang, D. Song, and D. Tao, "Hierarchical prototype networks for continual graph representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[38] H. Liu, Y. Yang, and X. Wang, "Overcoming catastrophic forgetting in graph neural networks," *arXiv preprint arXiv:2012.06002*, 2020.

[39] C. Wang, Y. Qiu, and S. Scherer, "Lifelong graph learning," *arXiv preprint arXiv:2009.00647*, 2020.

[40] Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, and M. Coates, "Graphsail: Graph structure aware incremental learning for recommender systems," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 2861–2868.

[41] A. Daruna, M. Gupta, M. Sridharan, and S. Chernova, "Continual learning of knowledge graph embeddings," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1128–1135, 2021.

[42] A. Rakaraddi, L. Siew Kei, M. Pratama, and M. De Carvalho, "Reinforced continual learning for graphs," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 1666–1674.

[43] Y. Han, S. Karunasekera, and C. Leckie, "Graph neural networks with continual learning for fake news detection from social media," *arXiv preprint arXiv:2007.03316*, 2020.

[44] A. Carta, A. Cossu, F. Errica, and D. Bacciu, "Catastrophic forgetting in deep graph networks: A graph classification benchmark," *Frontiers in artificial intelligence*, vol. 5, 2022.

[45] X. Kou, Y. Lin, S. Liu, P. Li, J. Zhou, and Y. Zhang, "Disentangle-based continual graph representation learning," *arXiv preprint arXiv:2010.02565*, 2020.

[46] A. Zaman, F. Yangyu, M. S. Ayub, M. Irfan, L. Guoyun, and L. Shiya, "Cmdgat: knowledge extraction and retention based continual graph attention network for point cloud registration," *Expert Systems with Applications*, p. 119098, 2022.

[47] M. Perini, G. Ramponi, P. Carbone, and V. Kalavri, "Learning on streaming graphs with experience replay," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 470–478.

[48] S. Efeoglu, "A continual relation extraction approach for knowledge graph completeness (short paper)," 2022.

[49] H. Bo, R. McConville, J. Hong, and W. Liu, "Ego-graph replay based continual learning for misinformation engagement prediction," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 01–08.

[50] L. Galke, I. Vagliano, and A. Scherp, "Incremental training of graph neural networks on temporal graphs under distribution shift," *arXiv preprint arXiv:2006.14422*, 2020.

[51] J. Wang, G. Song, Y. Wu, and L. Wang, "Streaming graph neural networks via continual learning," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1515–1524.

[52] W. Yu, W. Cheng, C. C. Aggarwal, K. Zhang, H. Chen, and W. Wang, "Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2672–2681.

[53] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Companion Proceedings of the The Web Conference 2018*, 2018, pp. 969–976.

[54] L. Zhou, Y. Yang, X. Ren, F. Wu, and Y. Zhuang, "Dynamic network embedding by modeling triadic closure process," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[55] Y. Ma, Z. Guo, Z. Ren, J. Tang, and D. Yin, "Streaming graph neural networks," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 719–728.

[56] F. Zhou, C. Cao, K. Zhang, G. Trajcevski, T. Zhong, and J. Geng, "Meta-gnn: On few-shot node classification in graph meta-learning," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2357–2360.

[57] Z. Guo, C. Zhang, W. Yu, J. Herr, O. Wiest, M. Jiang, and N. V. Chawla, "Few-shot graph learning for molecular property prediction," in *Proceedings of the Web Conference 2021*, 2021, pp. 2559–2567.

[58] H. Yao, C. Zhang, Y. Wei, M. Jiang, S. Wang, J. Huang, N. Chawla, and Z. Li, "Graph few-shot learning via knowledge transfer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6656–6663.

[59] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2020.

[60] X. Zhang, C. Xu, X. Tian, and D. Tao, "Graph edge convolutional neural networks for skeleton-based action recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 3047–3060, 2019.

[61] X. Ai, C. Sun, Z. Zhang, and E. R. Hancock, "Two-level graph neural network," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[62] X. Zhang, C. Xu, and D. Tao, "Context aware graph convolution for skeleton-based action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 333–14 342.

[63] X. Wu, Y. Yuan, X. Zhang, C. Wang, T. Xu, and D. Tao, "Gait phase classification for a lower limb exoskeleton system based on a graph convolutional network model," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 4999–5008, 2021.

[64] X. Zhang, C. Xu, and D. Tao, "On dropping clusters to regularize graph convolutional neural networks," in *European Conference on Computer Vision*, 2020.

[65] R. Forman, "Bochner's method for cell complexes and combinatorial ricci curvature," *Discrete and Computational Geometry*, vol. 29, no. 3, pp. 323–374, 2003.

[66] R. Sreejith, K. Mohanraj, J. Jost, E. Saucan, and A. Samal, "Forman curvature for complex networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2016, no. 6, p. 063206, 2016.

[67] Y. Lin, L. Lu, and S.-T. Yau, "Ricci curvature of graphs," *Tohoku Mathematical Journal, Second Series*, vol. 63, no. 4, pp. 605–627, 2011.

[68] W. Lin, H. Lan, and B. Li, "Generative causal explanations for graph neural networks," in *ICML*. PMLR, 2021, pp. 6666–6679.

[69] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He, "Graph information bottleneck for subgraph recognition," *arXiv preprint arXiv:2010.05563*, 2020.

[70] D. Bacciu and D. Numeroso, "Explaining deep graph networks via input perturbation," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[71] I. Spinelli, S. Scardapane, and A. Uncini, "A meta-learning approach for training explainable graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[72] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *International Conference on Learning Representations*, 2019.

[73] X. Zhang, D. Song, and D. Tao, "Cglb: Benchmark tasks for continual graph learning," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[74] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[75] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in neural information processing systems*, vol. 33, pp. 22 118–22 133, 2020.

[76] K. Ahrabian, Y. Xu, Y. Zhang, J. Wu, Y. Wang, and M. Coates, "Structure aware experience replay for incremental learning in graph-based recommender systems," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2832–2836.

[77] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6861–6871.

[78] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[79] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[80] G. Wu, S. Gong, and P. Li, "Striking a balance between stability and plasticity for class-incremental learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1124–1133.

[81] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.

[82] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4367–4375.

[83] Y. Yin, Q. Wang, S. Huang, H. Xiong, and X. Zhang, "Autogcl: Automated graph contrastive learning via learnable view generators," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 8, 2022, pp. 8892–8900.

[84] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 70–79.

[85] Z. Tong, Y. Liang, H. Ding, Y. Dai, X. Li, and C. Wang, "Directed graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19 580–19 593, 2021.

[86] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7793–7804, 2020.

[87] Y. Chen, H. Yang, Y. Zhang, K. Ma, T. Liu, B. Han, and J. Cheng, "Understanding and improving graph injection attack by promoting unnoticeability," *arXiv preprint arXiv:2202.08057*, 2022.

Xikun Zhang is Ph.D. student at the School of Computer Science in The University of Sydney. He obtained a M.phil from the School of Computer Science in The University of Sydney, and a Bachelor of Applied Physics from University of Science and Technology of China. His research interests mainly include applying deep learning to tackle graph related problems, and has publications in top-tier venues including CVPR, ECCV, TNNLS, TPAMI, NeurIPS, ICDM.

Dongjin Song (M'19) is an assistant professor in the Department of Computer Science and Engineering at the University of Connecticut. His research interests include machine learning, deep learning, time series analysis, graph representation learning, and related applications. Papers describing his research have been published at top-tier machine learning and artificial intelligence journals and conferences, such as T-PAMI, T-NNLS, T-IP, NeurIPS, ICML, ICLR, KDD, AAAI, IJCAI, CVPR, ICCV, ICDM, SDM, etc. He is an Associate Editor for Neurocomputing and regularly served as Senior PC for AAAI, IJCAI, and CIKM. He won the UConn Research Excellence Research (REP) Award in 2021. He has co-organized "IJCAI Workshop on Artificial Intelligence for Time Series Analysis (AI4TS)" and "KDD Workshop on Mining and Learning from Time Series" in the past 2 years (2022, 2023).

Dacheng Tao (IEEE Fellow) is currently a Professor of Computer Science, Peter Nicol Russell Chair and an Australian Laureate Fellow in the Sydney AI Centre and the School of Computer Science in the Faculty of Engineering at The University of Sydney. He was the founding director of the Sydney AI Centre. He mainly applies statistics and mathematics to artificial intelligence and data science, and his research is detailed in one monograph and over 200 publications in prestigious journals and proceedings at leading conferences. He received the 2015 and 2020 Australian Eureka Prize, the 2018 IEEE ICDM Research Contributions Award, and the 2021 IEEE Computer Society McCluskey Technical Achievement Award. He is a Fellow of the Australian Academy of Science, AAAS, ACM and IEEE.