Class-Domain Incremental Learning on Graphs via Disentangled Knowledge Distillation

Qin Tian
College of Intelligence and Computing, Tianjin University
Tianjin, China
tianqin123@tju.edu.cn

Chen Zhao
Department of Computer Science, Baylor University
Waco, Texas, USA
chen_zhao@baylor.edu

Xintao Wu
Department of Electrical Engineering and Computer Science, University of Arkansas
Fayetteville, Arkansas, USA
xintaowu@uark.edu

Dong Li
Department of Computer Science, Baylor University
Waco, Texas, USA
dong_li1@baylor.edu

Minglai Shao
School of New Media and Communication, Tianjin University
Tianjin, China
shaoml@tju.edu.cn

Xujiang Zhao
NEC Laboratories America
Princeton, New Jersey, USA
xuzhao@nec-labs.com

Wenjun Wang*
College of Intelligence and Computing, Tianjin University
Tianjin, China
wjwang@tju.edu.cn

Abstract

Graph incremental learning aims to sequentially adapt models to evolving graphs while mitigating catastrophic forgetting. This problem becomes particularly challenging due to the simultaneous occurrence of covariate and label distribution shifts, introduced by newly emerging node classes, changes in node feature styles, and additional edges. To address these challenges, we propose DINGLE, a novel framework for both class and domain (class-domain) incremental learning on graphs. DINGLE consists of two key modules: a representation decoupler, which disentangles node representations into domain-invariant semantic factors for classification and domain-specific variation factors, and a teacher-student knowledge distillation module, which facilitates knowledge transfer across tasks while mitigating catastrophic forgetting through memory replay. By leveraging a Representative Node Feature (RNF) bank and an Encoder Parameters (EP) bank, DINGLE ensures effective knowledge retention and adaptation. Extensive experiments on 5 real-world datasets demonstrate that DINGLE outperforms 11 state-of-the-art baselines in class-domain incremental learning, improving classification accuracy while effectively preventing forgetting across tasks.

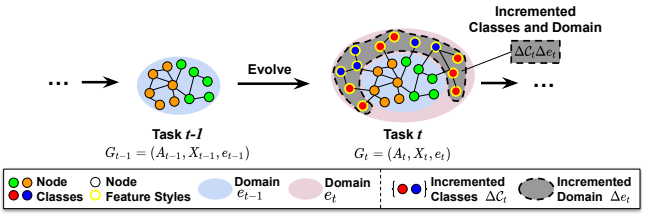

Figure 1: Illustration of class-domain incremental learning on expanding networks. At task t, the graph G_t expands upon the previous G_{t-1} by introducing new nodes and edges from novel classes and domains. This results in sequential changes in node representations, driven by both node features and topological structures, due to covariate shift, as well as changes in node classes due to label shift.

CCS Concepts

• **Computing methodologies** → *Online learning settings.*

Keywords

Incremental Learning, Graph Learning

ACM Reference Format:
Qin Tian, Chen Zhao, Xintao Wu, Dong Li, Minglai Shao, Xujiang Zhao, and Wenjun Wang. 2026. Class-Domain Incremental Learning on Graphs via Disentangled Knowledge Distillation. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792088

1 Introduction

Graph incremental learning, where each task arrives sequentially with a new graph, aims to continuously adapt the model to newly

*Wenjun Wang is the corresponding author and also in State Key Laboratory of Synthetic Biology and National Key Laboratory of Synthetic Biotechnology, Tianjin University, Tianjin, China, and Yazhou Bay Innovation Institute, Hainan Tropical Ocean University, Sanya Hainan, China.

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792088

emerging graphs while mitigating catastrophic forgetting from previous tasks. This approach has gained significant attention and has been applied in various fields [2, 6, 21, 25, 39, 42, 44]. For example, in a dynamically evolving citation network [44], where nodes represent papers and edges denote citation relationships, the network continuously expands as new papers from previously unseen topics (new classes) with an unseen writing style (distinct node feature styles) and their citations (additional edges) are added over time. As shown in Figure 1, this evolution results in changes to the graph domain, where a domain is defined as the distribution of node representations learned through graph neural networks (GNNs). These changes include shifts in the distribution of node features and classes, as well as modifications to the graph's topological structure. As a result, adapting the classifier learned from previous tasks to the expanded graph in the new task presents two main challenges: (1) The newly introduced nodes, with new classes, distinct node feature styles, and additional edges, result in the new graph exhibiting covariate and label distribution shifts compared to previous graphs. Covariate shift [27] refers to changes in the marginal distribution of node features and graph topologies, while label shift [35] denotes variations in the marginal distribution of node classes. (2) Another key challenge is ensuring that the classifier learned on the new graph maintains its ability to generalize across all previous graphs while preserving prediction accuracy for node classification, effectively mitigating catastrophic forgetting.

To tackle these challenges, several graph incremental learning methods have been proposed. Most of these methods focus exclusively on either domain-incremental learning [3, 14, 36, 38] (covariate shift) or class-incremental learning [15, 23, 41] (label shift), rather than addressing both simultaneously. A few approaches consider evolving graphs but treat covariate and class shift separately. They either ignore the change of node feature styles [33] or primarily focus on optimizing GNNs [5] which fail to decouple graph variations from semantic information and lack mechanisms for continual adaptation to both novel domains and emerging classes. As a result, these methods struggle to adapt to nodes with new classes emerging from new domains.

Furthermore, existing methods designed to address catastrophic forgetting in graph incremental learning can be broadly categorized into three approaches. Parameter isolation methods [40] dynamically allocate new parameters to adapt to different tasks while partially or entirely isolating existing ones. Regularization-based methods [23] impose constraints on critical parameters from previous tasks to prevent drastic changes, thereby preserving prior knowledge. Memory replay-based methods [26, 29, 33, 41] store representative data from past tasks and replay them during new task learning. However, under covariate and label shifts, these methods struggle to effectively prevent catastrophic forgetting and exhibit low node classification accuracy on graphs from previous tasks. This is because they fail to preserve the key semantics of previous tasks, ultimately misguiding the learning process.

In this paper, we propose DINGLE, a novel framework designed to address the challenges of class-domain incremental learning on graphs, handling sequential tasks under both label shift and covariate shift simultaneously. Specifically, DINGLE introduces two key modules: a representation decoupler and a teacher-student knowledge distillation module. The representation decoupler disentangles node representations learned from GNNs into semantic and variation factors, where semantic factors are used for node classification, while variation factors capture domain-specific information. The knowledge distillation module comprises three key components: bank construction, memory replay, and teacher-student distillation. We initialize a representative node feature (RNF) bank, which sequentially stores selected node representations for each class, and an encoder parameters (EP) bank, which preserves the learned parameters of semantic and variation encoders from previous tasks. During memory replay, the teacher model generates semantic and variation factors by leveraging both banks, where its parameters are weighted using those stored in the EP bank. To mitigate catastrophic forgetting, knowledge distillation minimizes the distance between semantic factors while maximizing the dissimilarity of variation factors between the teacher and student models. This enhances factor disentanglement in the student model and improves classification accuracy for node classification. Our contributions are summarized as follows.

- We introduce a class-domain incremental learning problem on graphs that sequentially considers both class-incremental and domain-incremental learning in graph neural networks. This problem explicitly focuses on covariate shifts, arising from evolving node features and graph structures, as well as label shifts, where new classes emerge from distinct domains, simultaneously.
- We present DINGLE, a novel framework that integrates representation disentanglement and knowledge distillation, where the representation decoupler separates semantic factors from variations, improving generalization under distribution shifts, and the student-teacher distillation module facilitates knowledge transfer across tasks while mitigating catastrophic forgetting.
- The effectiveness and efficiency of DINGLE are evaluated on 5 real-world datasets, demonstrating superior performance over 11 state-of-the-art baselines. The results demonstrate that DINGLE enhances classification accuracy in class-domain incremental learning settings while effectively mitigating catastrophic forgetting across tasks.

2 Related Work

Incremental learning on graphs aims to mitigate catastrophic forgetting while enabling models to adapt to newly emerging graphs on streaming graph data, which has garnered significant attention [4, 5, 9, 18, 23, 31, 33, 39, 42]. Based on task identity availability during testing, graph incremental learning is categorized into three settings [31, 44]: task-incremental learning, class-incremental learning, and domain-incremental learning.

Task-Incremental Learning on Graphs. In task-incremental learning, the model is aware of the task identity during testing, with each task containing distinct classes [17, 28, 43, 45]. Consequently, the model only needs to distinguish between classes within the current task. For example, ER-GNN [45] stores representative node features from past tasks and replays them during training to retain historical knowledge. Other approaches, such as TWP [17], SSM [43], focus on preserving the topological structure of graphs across tasks to maintain relational information.

Class-Incremental Learning on Graphs. It aims to develop a unified classifier capable of recognizing an ever-expanding set of

classes over time, which is extensively studied [23, 26, 31, 34, 41]. HAG-Meta [29] employs task and node level attention regularization to refine prototype representations, ensuring both knowledge retention and adaptability with limited labeled data. Further, Geometer [23] refines attention-based prototypes to predict node labels while preserving observed class information through knowledge distillation. PDGNNs [41] leverages memory replay to retain key semantic information and mitigate forgetting.

Domain-Incremental Learning on Graphs. It aims to develop a classifier that generalizes across tasks with significant distribution shifts [3, 14, 30, 36, 38]. DiCGRL [14] uses attention mechanisms to disentangle information that defines node relationships, enabling adaptation to streaming multi-relational data. CI-LightGCN [3] incorporates GNN guided by causal principles and employs distillation to mitigate outdated node representations. Currently, research on domain-incremental learning for graph data is limited.

Most existing methods focus solely on domain-incremental (covariate shift) or class-incremental learning (label shift), neglecting their joint impact. This paper tackles class-domain incremental learning, considering both in graph neural networks.

3 Problem Formulation

In this section, we provide the problem formulation for class-domain incremental learning on graphs. Given a stream of graphs $\mathcal{G} = \{G_t\}_{t=1}^{T}$, where each graph G_t arrives sequentially at task $t \in [T]$ and expands from G_{t-1} by introducing additional nodes and edges from novel classes and domains. $[T]$ denotes sequential tasks $\{1, \cdots, T\}$. Each graph $G_t = (A_t, X_t, e_t)$ includes an adjacency matrix $A_t \in \{0,1\}^{|\mathcal{V}_t| \times |\mathcal{V}_t|}$, a d-dimensional node feature matrix $X_t = \{\mathbf{x}_{t,i}\}_{i=1}^{|\mathcal{V}_t|} \in \mathbb{R}^{|\mathcal{V}_t| \times d}$, where $\mathcal{V}_t$ is the set of nodes, and a domain indicator e_t specific to G_t. e_t indicates the distribution of node representations learned from GNNs. In this paper, we narrow the scope to distribution shifts across tasks, which arise from changes in the covariate distribution [27] of node features and graph topological structure, as well as label shifts [35] due to incremental node classes. We denote $\mathbf{y}_t = \{y_{t,i}\}_{i=1}^{|\mathcal{V}_t|} \in \mathbb{R}^{|\mathcal{V}_t|}$ as the node labels in G_t. Additionally, we denote the sets of classes across the streaming tasks as $\{C_t\}_{t=1}^{T}$ and the sets of domains as $\{e_t\}_{t=1}^{T}$, corresponding to the task sequence $[T]$. As shown in Figure 1, at the t-th streaming task, $|\Delta C_t|$ novel classes are introduced from a unique novel domain Δe_t, where $\Delta C_t = C_t \backslash C_{t-1}$, and for all tasks $\Delta C_i \cap \Delta C_j = \emptyset$ and $\Delta e_i \neq \Delta e_j$, $\forall i, j \in [T]$.

Throughout this paper, uppercase symbols represent variables and matrices, bold lowercase letters denote vectors, regular lowercase letters indicate scalars, and calligraphic symbols represent sets. A detailed list of notations used in this paper is provided in Table 3 in Appendix A.

Problem 1 (Class-Domain Incremental Learning on Graphs). *Given a sequence of tasks $[T] := \{1, \cdots, T\}$ where each task t arrives sequentially with a graph $G_t = (A_t, X_t, e_t)$ in domain e_t, $\forall t \in [T]$ and its node labels $\mathbf{y}_t$ corresponding to a set of classes C_t, a graph G_{t+1} at task $t+1$ is incrementally expanded form G_t with additional nodes and edges. These newly introduced nodes belong to novel classes ΔC_{t+1} and exhibit new node feature styles distinct from those in G_t. As a result, the additional nodes and edges lead to a different domain for G_{t+1} in e_{t+1}. The objective of this problem is to incrementally learn a classifier $f : \mathcal{A} \times \mathcal{X} \to \mathcal{Y}$ from the task sequence. At task t, the classifier f is required to minimize the empirical loss effectively and efficiently for node classification on G_t, while adapting to newly introduced classes and domains that were unseen from previous tasks. Additionally, it must preserve the accuracy of node classification on previously encountered graphs $\{G_i\}_{i=1}^{t-1}$.*

To address Problem 1, two key challenges must be considered: (1) Each graph G_t introduces additional nodes and edges with new node classes and domain variations, making it difficult to adapt previously learned classifiers to G_t. (2) Ensuring that the classifier f learned at task t retains the ability to make accurate predictions for known node classes in previous graphs $\{G_i\}_{i=1}^{t-1}$ is crucial, as failure to do so can lead to catastrophic forgetting. To overcome these two challenges, we propose DINGLE, which is detailed in the following section.

4 Methodology

An overview of DINGLE is provided in Figure 2, where it leverages two key techniques: node representation disentanglement to address the first challenge and knowledge distillation combined with memory replay to tackle the second challenge. Specifically, the representation decoupler separates node representations into domain-invariant semantic factors, which serve as inputs for classification, and domain-specific variation factors, which are class-independent. This disentanglement enables accurate predictions under distribution shifts (Section 4.1). Additionally, knowledge distillation facilitates semantic knowledge transfer across tasks through a memory replay mechanism that stores key nodes and model parameters from previous tasks, effectively mitigating catastrophic forgetting (Section 4.2).

4.1 Representation Decoupler

Although GNNs are effective in capturing key information for prediction, node representations they learn are often mixed with domain-specific environmental features. These domain-specific features change dynamically with domain shifts. When introducing nodes from new domains, GNNs may mislead important semantic information from previous tasks. To address this, following [11], we assume that node representations learned through GNNs can be disentangled into domain-invariant semantic factors, which are further used for class prediction, and domain-specific variation factors, which are class-irrelevant.

Given a graph G_t, a classifier f at task t includes a node representation function $\text{GNN}_t : \mathcal{A}_t \times \mathcal{X}_t \to \mathbb{R}^{|\mathcal{V}_t| \times n}$, a semantic encoder $E_t^c : \mathbb{R}^n \to \mathbb{R}^{d_c}$ parameterized by $\boldsymbol{\theta}_t^c$, a variation encoder $E_t^v : \mathbb{R}^n \to \mathbb{R}^{d_v}$ parameterized by $\boldsymbol{\theta}_t^v$, a decoder $D_t : \mathbb{R}^{d_c} \times \mathbb{R}^{d_v} \to \mathbb{R}^n$, and a classification head $h_t : \mathbb{R}^{d_c} \to \mathcal{Y}_t$. For simplicity, we omit the subscripts t in the notation throughout the remainder of this section. We define the latent semantic factor as $\mathbf{c} = E^c(\mathbf{z})$ and the latent variation factor as $\mathbf{v} = E^v(\mathbf{z})$, where $\mathbf{z}$ is a node representation from $Z = \text{GNN}(A, X)$. Specifically, we disentangle semantic and variation factors by training the encoders and decoder, leveraging bidirectional reconstruction losses to reconstruct both node representations and latent factors. For node representation reconstruction, we use the encoders E^c, E^v and decoder D to minimize the loss between the original node representations learned through

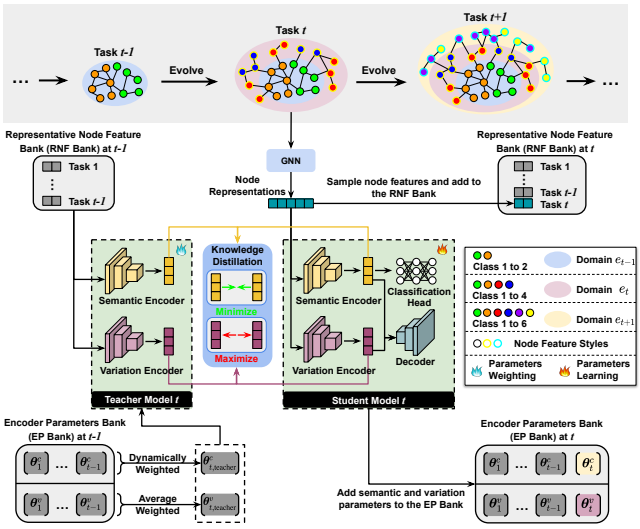

Figure 2: Overview of the proposed DINGLE. At task t, the student model disentangles node representations into semantic and variation factors, while the teacher generates corresponding factors using the RNF and EP banks from task $t-1$. Knowledge distillation minimizes semantic discrepancy and maximizes variation dissimilarity to alleviate forgetting. At the end of task t, representative node features are selected and added to the RNF bank, while the encoder parameters are stored in the EP bank for future use.

GNNs and the reconstructed representations generated from the encoded semantic $\mathbf{c}$ and variation factors $\mathbf{v}$. The reconstruction loss $\mathcal{L}^z_{rec}$ can be formulated as follows:

$$\mathcal{L}^z_{rec} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\left[\left\|D(\mathbf{c}_i, \mathbf{v}_i) - \mathbf{z}_i\right\|_1\right]. \tag{1}$$

For factor reconstruction, we randomly sample the variation factors $\hat{\mathbf{v}}$ from the prior distribution $\mathcal{N}(0, \mathbf{I})$. The sampled $\hat{\mathbf{v}}$ and the encoded semantic factor $\mathbf{c}$ are used to generate pseudo node representations $\mathbf{z}' = D(\mathbf{c}, \hat{\mathbf{v}})$ with diverse variations. $\mathbf{z}'$ is further encoded into a latent semantic factor $\mathbf{c}' = E^c(\mathbf{z}')$ and a latent variation factor $\mathbf{v}' = E^v(\mathbf{z}')$. Ideally, $\mathbf{c}$ and $\mathbf{c}'$ should be identical, and similarly, $\hat{\mathbf{v}}$ and $\mathbf{v}'$ should be closely matched. Therefore, we introduce the factor reconstruction losses as follows:

$$\mathcal{L}^c_{rec} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\left[\left\|E^c(D(\mathbf{c}_i, \hat{\mathbf{v}}_i)) - \mathbf{c}_i\right\|_1\right], \tag{2}$$

$$\mathcal{L}^v_{rec} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\left[\left\|E^v(D(\mathbf{c}_i, \hat{\mathbf{v}}_i)) - \hat{\mathbf{v}}_i\right\|_1\right]. \tag{3}$$

Ensuring that the generated pseudo node representations are indistinguishable from the representations learned from GNNs is crucial. To achieve this, we introduce a discriminator Φ: $\mathbb{R}^n \to \mathbb{R}$ and the discrimination loss $\mathcal{L}_{dis}$ is defined:

$$\mathcal{L}_{dis} = -\frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\log\Phi(\mathbf{z}_i) - \frac{1}{M}\sum_{j=1}^{M}\log(1-\Phi(\mathbf{z}'_j)), \tag{4}$$

where M represents the number of $\mathbf{z}'$ with random variations $\hat{\mathbf{v}}$. The first term maximizes the probability of correctly classifying $\mathbf{z}$ and the second term minimizes the probability of misclassifying $\mathbf{z}'$. Finally, we jointly train all encoders, the decoder, and the discriminator:

$$\mathcal{L}_{Decouple} = \mathcal{L}_{dis} + \lambda_z\mathcal{L}^z_{rec} + \lambda_c\mathcal{L}^c_{rec} + \lambda_v\mathcal{L}^v_{rec}, \tag{5}$$

where $\lambda_z, \lambda_c, \lambda_v > 0$ are hyper-parameters. In addition, the classification head takes semantic factors as input for node class prediction. The classification loss is formulated:

$$\mathcal{L}_{CE} = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\ell\Big(h(E^c(\mathbf{z}_i)), y_i\Big), \tag{6}$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the cross entropy loss.

4.2 Teacher-Student Knowledge Distillation

As new classes are introduced, the model's ability to retain knowledge from previous tasks gradually declines, leading to the issue of "forgetting old". Additionally, domain shifts between new and old tasks further exacerbate this problem, as training on the new task causes the model to overlook critical or shared knowledge from previous tasks. To address these challenges, we design a knowledge distillation module that distills key semantic information from previous tasks $[t-1]$ and transfers it to task t to facilitate cross-domain knowledge sharing. This module consists of three main components:

- **Bank Construction**: We initialize two banks, a Representative Node Feature (RNF) Bank and an Encoder Parameters (EP) Bank. The RNF Bank stores representative node features for each class across all previous tasks, while the EP Bank preserves the parameters $\{\boldsymbol{\theta}^c_i\}_{i=1}^{t-1}$ and $\{\boldsymbol{\theta}^v_i\}_{i=1}^{t-1}$ of both previous semantic and variation encoders.
- **Memory Replay**: At task t, the teacher model, consisting of a semantic encoder E^c_{tcher} and a variation encoder E^v_{tcher}, processes all representative node features stored in the RNF bank from task $t-1$. It outputs experienced semantic and variation factors using weighted parameters $\boldsymbol{\theta}^c_{t,\text{tcher}}$ and $\boldsymbol{\theta}^v_{t,\text{tcher}}$, which are estimated from the EP Bank at $t-1$.
- **Teacher-Student Distillation**: To facilitate effective knowledge transfer, we aim to minimize the distance between the corresponding semantic factors while maximizing the distance between the corresponding variation factors across the teacher and student models.

Bank Construction. An RNF bank and an EP bank are initialized at each task. Specifically, at the end of task $t-1$, we store the parameters of the semantic encoder $\boldsymbol{\theta}^c_{t-1}$ and variation encoder $\boldsymbol{\theta}^v_{t-1}$ in the EP Bank. Since the domain variation of a graph G_t is determined by its node features and the topological structure, we aim to reduce computational complexity while retaining as much of the nodes' attribute and topological information as possible. Inspired by [45], we select the most representative node features for each class and store them in the RNF Bank. To identify these representative node features, we compute K_t prototypes for all classes, where K_t denotes the number of classes at t. Each prototype is obtained by averaging the node representations within its respective class

Algorithm 1 The procedure of DINGLE

Input: A sequence of graphs $\{G_t\}_{t=1}^{T}$, where each graph at task t includes additional nodes and edges with novel node classes from a new domain
Initialization: An RNF and an EP banks
for each task t **do**
  Compute teacher model parameters $\boldsymbol{\theta}^{c}_{t,\text{tcher}}, \boldsymbol{\theta}^{v}_{t,\text{tcher}}$ using the EP bank following Eqs. (8) and (9)
  Generate teacher model factors using the RNF bank
  repeat
    Decouple node representations of G_t in to $\mathbf{c}_{t,\text{stud}}, \mathbf{v}_{t,\text{stud}}$ in student model
    Compute the decoupling loss $\mathcal{L}_{Decouple}$ and node classification loss $\mathcal{L}_{CE}$ using Eqs. (5) and (6)
    Compute teacher-student loss $\mathcal{L}_{KD}$ using Eq. (10)
    Minimize the total loss $\mathcal{L}_{Decouple} + \mathcal{L}_{CE} + \mathcal{L}_{KD}$
  until convergence
  Save student model parameters $\boldsymbol{\theta}^{c}_{t,\text{stud}}, \boldsymbol{\theta}^{v}_{t,\text{stud}}$ in the EP bank
  Select representative node features from node representations and save them in the RNF bank
end for

$k \in \{1, \cdots, K\}$.

$$\bar{\mathbf{z}}_{k,t} = \frac{1}{|\mathcal{Z}_{k,t}|} \sum_{\mathbf{z}_i \in \mathcal{Z}_{k,t}} \mathbf{z}_i, \tag{7}$$

where $\mathcal{Z}_{k,t}$ is the set of node representations of class k. $\mathbf{z}_i$ is a node representation learned from GNN. We then select the top m node representations closest to each prototype $\bar{\mathbf{z}}_{k,t}$. Finally, $K \times m$ node representations are added to the RNF Bank.

Memory Replay. At task t, the RNF bank stores $K \times m \times (t-1)$ representative node features with varying numbers of features per class. For example, if $|\Delta C_t| = 2$, class 1 and 2 each have $t-1$ features, class 3 and 4 each have $t-2$ features, and so on. This leads to repeated learning of the same classes across different tasks. To preserve previous semantic information while preventing overemphasis on the same classes, we estimate the parameters of the teacher model's semantic and variation encoders by weighting the stored parameters from the previous $t-1$ tasks in the EP Bank.

$$\boldsymbol{\theta}^{c}_{t,\text{tcher}} = \sum_{i=1}^{t-1} \omega_c \cdot \boldsymbol{\theta}^{c}_{i}, \quad \text{where} \quad \omega_c = \frac{t-i}{\sum_{j=1}^{t-1} j} \tag{8}$$

$$\boldsymbol{\theta}^{v}_{t,\text{tcher}} = \sum_{i=1}^{t-1} \omega_v \cdot \boldsymbol{\theta}^{v}_{i}, \quad \text{where} \quad \omega_v = \frac{1}{t-1}, \tag{9}$$

As shown in Figure 2, unlike the student model whose parameters are learned during training, $\boldsymbol{\theta}^{c}_{t,\text{tcher}}$ and $\boldsymbol{\theta}^{v}_{t,\text{tcher}}$ in the teacher model are estimated using the EP bank. Taking node features in the RNF bank as inputs, the teacher model outputs semantic factors $\mathbf{c}_{t,\text{tcher}}$ and variation factors $\mathbf{v}_{t,\text{tcher}}$.

Teacher-Student Distillation. To enable the student model to inherit the classification knowledge of previously learned classes and enhance the disentanglement of semantic and variation factors, we employ teacher-student knowledge distillation by regulating the discrepancy between these corresponding factors in both the student and teacher models. We aim to minimize the distance between semantic factors while maximizing the dissimilarity between variation factors across the two models. The teacher-student distillation loss $\mathcal{L}_{KD}$ can be defined as:

$$\mathcal{L}_{KD} = \sum_{k=1}^{|C_{t-1}|} \Big(\text{Dist}(\mathcal{B}^{c}_{t,\text{tcher},k}, \mathcal{B}^{c}_{t,\text{stud},k}) - \text{Dist}(\mathcal{B}^{v}_{t,\text{tcher},k}, \mathcal{B}^{v}_{t,\text{stud},k}) \Big), \tag{10}$$

where $\text{Dist}(\cdot,\cdot)$ is a distance metric. Here, $\mathcal{B}^{c}_{t,\text{tcher},k}$ and $\mathcal{B}^{c}_{t,\text{stud},k}$ represent the sets of semantic factors for the k-th class seen before task t in the teacher and student models, respectively. Similarly, $\mathcal{B}^{v}_{t,\text{tcher},k}$ and $\mathcal{B}^{v}_{t,\text{stud},k}$ denote the sets of variation factors for the k-th old class in the teacher and student models, respectively.

The teacher-student knowledge distillation module helps avoid catastrophic forgetting by transferring knowledge from old tasks while accommodating the addition of novel classes from new domains. We provide the whole process of DINGLE in Algorithm 1.

Table 1: Key Characteristics of Datasets

Dataset	# Nodes	# Edges	Node Dimension	# Classes (Base)	# Tasks	# Incremental Classes
CoraFull [24]	19,793	126,842	8,710	70 (20)	11	5
CoraML [1]	2,995	16,316	2,879	7 (2)	6	1
Reddit [8]	232,965	114,615,892	602	40 (20)	11	2
Products [10]	2,449,029	61,859,140	100	47 (27)	11	2
Amazon [37]	13,752	491,722	767	10 (2)	9	1

5 Experiments

We conduct extensive experiments on the proposed DINGLE across a variety of five real-world datasets to address the following research questions (RQs):

- **RQ1:** How does the overall performance of DINGLE across tasks in class-domain incremental learning compare to state-of-the-art methods?
- **RQ2:** How effective is DINGLE in node representation disentanglement and mitigating catastrophic forgetting?
- **RQ3:** How do the key components contribute to the effectiveness of DINGLE?

5.1 Experiment Settings

Datasets. We evaluate the proposed DINGLE on five datasets: CoraFull [24], CoraML [1], Reddit [8], Products [10], and Amazon [37]. The key characteristics are summarized in Table 1. Each dataset is divided into a base task (Task 0) and a series of incremental tasks that arrive sequentially. The base task contains several classes from one domain, while each incremental task introduces new classes (incremental classes) from distinct domains (incremental domain), thus introducing both a growing label space and covariate shifts. The definition of domain follows the protocol introduced by [7], which has been widely used in graph learning studies [19, 20] to evaluate model robustness under distribution shifts. More details about the datasets and task construction are provided in Appendix B.

Table 2: Performance comparisons under class-domain incremental learning on different datasets using the GCN backbone. Results on other backbones, GAT and GraphSAGE, refer to Table 4 and Figure 9.

Backbone	Methods	CoraFull		CoraML		Reddit		Products		Amazon	
		AA/% ↑	AF/% ↑	AA/% ↑	AF/% ↑	AA/% ↑	AF/% ↑	AA/% ↑	AF/%↑	AA/% ↑	AF/%↑
GCN [12]	LWF [16]	38.7 ± 3.6	-40.0 ± 5.1	38.5 ± 4.7	-64.7 ± 5.3	39.0 ± 5.0	-41.5 ± 2.5	31.7 ± 4.2	46.0 ± 2.9	33.5 ± 1.9	-30.6 ± 3.1
	EWC [13]	40.6 ± 4.2	-42.3 ± 3.3	43.4 ± 2.1	-33.2 ± 3.9	38.3 ± 3.6	-40.1 ± 4.1	29.9 ± 1.8	-43.9± 5.0	30.5 ± 4.2	-38.0 ± 4.8
	TWP [17]	40.1 ± 3.9	-45.2 ± 3.4	47.2 ± 3.0	-36.0 ± 4.4	41.1 ± 2.8	-29.7 ± 3.5	30.6 ± 4.7	-44.2 ± 4.4	27.1 ± 3.7	-34.2 ± 2.6
	ER-GNN [45]	49.9 ± 3.3	-29.9 ± 3.6	50.9 ± 3.8	-26.7 ± 4.7	49.6 ± 4.0	-25.0 ± 4.5	45.8 ± 2.8	-34.7 ± 2.9	52.8 ± 1.6	-20.7 ± 2.0
	SSM [43]	50.7 ± 3.1	-27.8 ± 2.7	41.9 ± 2.8	-50.2 ± 5.9	47.9 ± 4.3	-22.2 ± 4.6	46.0 ± 4.1	-30.6 ± 3.0	48.6 ± 2.0	-26.7 ± 2.2
	SSRM [28]	53.9 ± 2.9	-23.3 ± 4.0	50.8 ± 3.3	-46.6 ± 4.8	52.0 ± 5.6	-28.3 ± 2.7	51.4 ± 3.1	-28.6 ± 1.5	55.3 ± 2.0	-22.1 ± 1.6
	LLGNN [5]	44.1 ±3.2	-40.5 ± 6.9	43.3 ± 4.7	-56.2 ± 5.1	43.6 ± 2.4	-33.7 ± 3.0	39.6 ± 3.8	-47.9 ±2.8	42.4 ± 4.7	-30.7 ± 4.8
	FGN [33]	52.2 ± 1.7	-25.1 ± 2.2	49.5 ± 1.8	-49.4 ± 2.8	50.3 ± 2.1	-27.8 ± 3.2	44.7 ± 3.7	-38.2 ± 2.5	49.9 ± 1.7	-22.6 ± 2.8
	PDGNNs [41]	60.2 ± 3.0	-10.0 ± 3.6	72.3 ± 2.9	-7.6 ± 3.0	85.3 ± 4.1	-11.5 ± 3.8	63.3 ± 2.8	**-10.7 ± 2.1**	69.2 ± 2.1	**-9.3 ± 1.9**
	GSIP [15]	53.2 ± 2.1	-19.9 ± 2.9	69.8 ± 0.9	-10.7 ± 2.0	76.4 ± 1.8	-26.4 ± 1.5	59.9 ± 2.7	-18.1 ± 3.0	56.4 ± 3.0	-20.0 ± 1.8
	Geometer [23]	57.1 ± 2.8	-20.1 ± 3.0	61.7 ± 2.3	-14.6 ± 2.7	49.2 ± 4.9	-22.1 ± 4.4	66.2 ± 5.1	-29.8 ± 3.7	61.3 ± 3.0	-16.6 ± 2.8
	DINGLE (Ours)	**63.6 ± 1.4**	**-4.8 ± 1.9**	**78.8 ± 2.0**	**-3.3 ± 1.3**	**89.5 ± 0.9**	**-8.9 ± 0.6**	**65.9±1.5**	-11.3 ± 1.1	**70.9 ± 1.8**	-9.9 ± 1.6

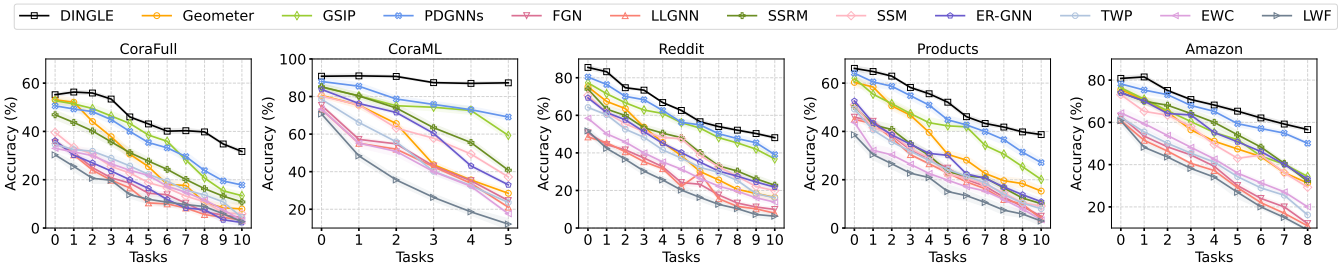

Figure 3: Learning dynamics of each task on all datasets.

Competing Baselines. We compare the performance of DINGLE with 11 baselines that fall into three categories: (1) 3 state-of-the-art class incremental learning approaches, including Geometer [23], GSIP [15], and PDGNNs [41], (2) 2 methods designed for class and domain incremental learning simultaneously, including FGN [33] and LLGNN [5], (3) 6 state-of-the-art task incremental learning methods adapted to the setting of class-domain incremental learning, including SSRM [28], SSM [43], ER-GNN [45], TWP [17], and 2 baselines for Euclidean data but also applicable to graphs, EWC [13], LWF [16]. The details are shown in Appendix C.

Evaluation Metrics. Following [15, 22, 28, 41, 43], we evaluate the model's performance using two widely adopted metrics: Average Accuracy (AA) and Average Forgetting (AF). AA reflects the learning dynamics across all learned tasks after completing each task, while AF measures the average performance degradation on previous tasks after learning a new task. Specifically, to compute these metrics, we construct an accuracy matrix $W^{Acc}_{t,i} \in \mathbb{R}^{T\times T}$ representing the accuracy on task i after learning task $t, \forall t \geq i$, with T denoting the total number of tasks.

$$\text{AA} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{t}\frac{W^{Acc}_{t,i}}{t}, \quad \text{where } t = 1,\ldots,T$$

$$\text{AF} = \frac{1}{T}\sum_{t=1}^{T}\frac{\sum_{i=1}^{t-1}(W^{Acc}_{t,i} - W^{Acc}_{i,i})}{t-1}, \quad \text{where } t = 2,\ldots,T$$

Implementation Details. We use a two-layer GCN [12] with a hidden dimension of 256 as the backbone in DINGLE. Additionally, we employ a two-layer GAT [32] with the same hidden dimension of 256. For GraphSAGE [8], we use a two-layer neural network with a hidden dimension of 256 and set the aggregation function to the mean. Both E^c and E^v are instantiated as a multi-layer perceptron (MLP) with one hidden layer of 128 neurons. The classifier is implemented as a linear layer. The results are reported as the average performance over 10 runs.

Hyperparameter Tuning. The hyper-parameters of the representation decoupler include some weight of loss $\lambda_z, \lambda_c, \lambda_v$. We chose them from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. Different combinations are adjusted for different datasets to achieve optimal performance. The hyper-parameters of the teacher-student knowledge distillation include m (the number of sampled nodes in RNF Bank), which is set in $\{1, 5, 10, 15, 20, 25, 30\}$. The number of iterations is 2000. For the hyper-parameters in learning the GNNs, the learning rate is chosen from $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$.

5.2 Experiment Results

Overall performance of DINGLE. We evaluate the performance of all methods on five real-world datasets. The results are presented in Table 2, with bold numbers indicating the best performance and underlined numbers representing the runner-up. The learning

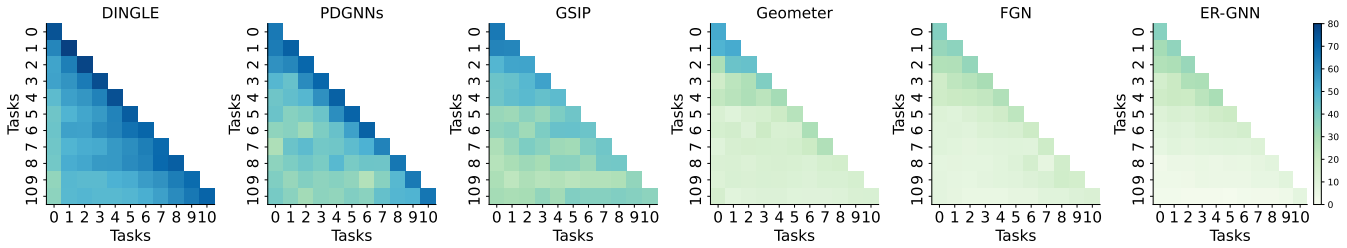

Figure 4: Accuracy matrix of DINGLE (Ours), PDGNNs [41], GSIP [15], Geometer [23], FGN [33], and ER-GNN [45] on the CoraFull dataset. Each cell in a heatmap represents the classification accuracy, where the cell at coordinate (x, y) indicates the accuracy on task x after learning task y. A darker cell indicates higher accuracy.

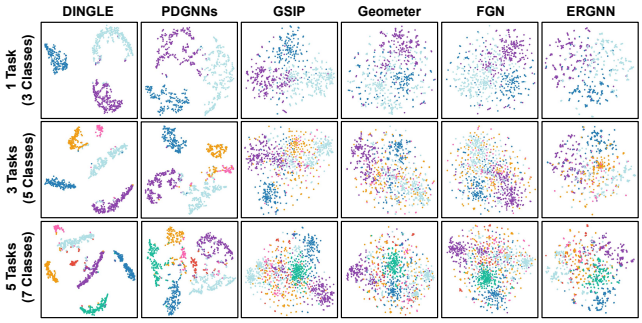

Figure 5: t-SNE visualization of node classifier inputs of DINGLE (Ours), PDGNNs [41], GSIP [15], Geometer [23], FGN [33], and ER-GNN [45] on the CoraML dataset after learning the 1st, 3rd, and 5th task. Different colors represent different classes.

dynamics of each task are illustrated in Figure 3, from which we draw the following observations.

Firstly, in the class-domain incremental learning setting, Table 2 shows that DINGLE outperforms baselines across various datasets, achieving better AA and AF (3% and 5.2% in CoraFull, 6.5% and 4.3% in CoraML, 4.2% and 2.6% in Reddit, 2.6% of AA in Products, and 1.7% of AA in Amazon). Additionally, as shown in Figure 3, DINGLE exhibits minimal accuracy degradation compared to the baselines as the number of tasks increases, highlighting the superiority of our approach. This can be attributed to three key reasons: (1) The encoders in DINGLE disentangle node representations into domain-invariant semantic factors and domain-specific variation factors, effectively mitigating the impact of domain shifts and enhancing the model's ability to capture semantic information for class prediction. (2) Unlike existing knowledge distillation techniques designed for class incremental learning, the teacher model in DINGLE does not directly inherit parameters from the previous task. Instead, it integrates weighted parameters from the semantic and variation encoders of all previous tasks. (3) The knowledge distillation term $\mathcal{L}_{KD}$ between teacher and student models reinforces disentanglement and preserves semantic information from earlier tasks. Together, these mechanisms effectively mitigate catastrophic forgetting in the class-domain incremental learning setting.

Secondly, among the baselines, PDGNNs ranks second to DINGLE. It adopts memory replay by storing low dimensional representations of key nodes from previous tasks and replaying them in later tasks to preserve topology. However, its parameter decoupled aggregation of neighbor attributes can also absorb domain specific variations that are unrelated to labels, which degrades prediction for new classes from new domains. Several baselines share this limitation, while representation disentanglement in DINGLE alleviates it. Geometer's teacher model inherits parameters from the previous task, but as new classes appear it suffers from node imbalance and fails to learn classes far from the current task. DINGLE addresses this by reweighting parameters from both semantic and variation encoders across tasks, ensuring balanced learning while preserving key knowledge. FGN and LLGNN consider class and domain incremental settings, yet they still integrate task irrelevant information and lack the ability to retain and transfer knowledge from old classes, which harms performance on new class nodes from new domains.

t-SNE visualization of node classifier inputs across tasks. To better understand the learning process of DINGLE, we visualize inputs of the node classifier from different classes using t-SNE while learning a sequence of 6 tasks on the CoraML dataset. We compare DINGLE with 5 baselines, including PDGNNs, GSIP, Geometer, FGN, and ER-GNN, for tasks 1 (3 classes), 3 (5 classes), and 5 (7 classes), as shown in Figure 5. Specifically, the input features of the classifier of DINGLE correspond to the outputs of the semantic encoder for node representations. As observed, with the introduction of new classes from different domains, DINGLE effectively clusters nodes with the same label across all tasks. This can be attributed to feature disentanglement, which separates semantic and variation factors, reducing the model's reliance on domain variations that are unrelated to class labels and prone to dynamic changes. Additionally, teacher-student distillation and parameter re-weighting enhance the model's ability to preserve and learn from knowledge across all previous tasks.

The effectiveness of DINGLE in catastrophic forgetting. To gain a comprehensive understanding, we visualize the accuracy matrices of DINGLE, PDGNNs, GSIP, Geometer, FGN, and ER-GNN on the CoraFull dataset across all tasks in Figure 4. Each cell (t_x, t_y) in a heatmap represents the predicted accuracy of the model trained on task t_x when evaluated on task t_y task. Darker blue indicates higher accuracy. Compared to the baselines, DINGLE consistently maintains stable performance across tasks, even as new classes from different domains are continuously learned.

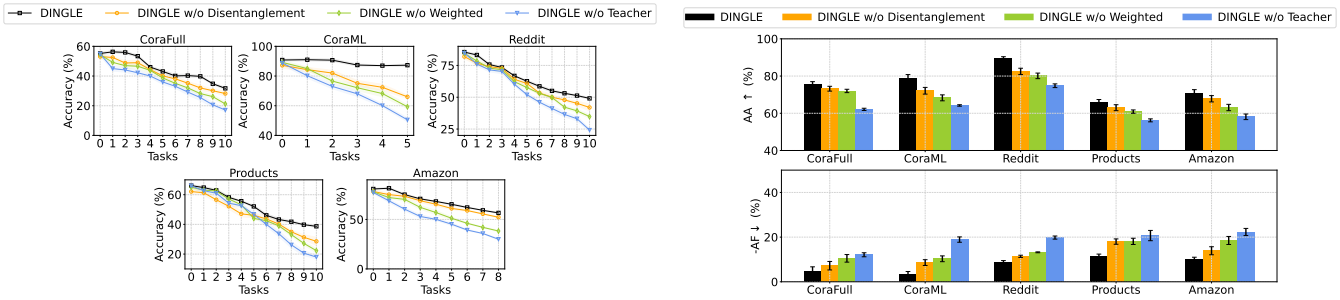

Figure 6: Ablation study results on all datasets. (Left) Node classification accuracy across tasks. (Right) Results of Average Accuracy (AA) and Average Forgetting (AF).

Ablation studies. We investigate the effectiveness of each key module in DINGLE, including the representation encoders (w/o Disentanglement), parameter weighting (w/o Weighted), and teacher-student knowledge distillation (w/o Teacher), in all datasets, using the full model DINGLE as the standard for the experiments. The results are shown in Figure 6. (1) In DINGLE w/o Disentanglement, we remove encoders and use a GNN in both the student and teacher models to learn node representations for classification. We observe a 2%-7% drop in AA and a 3%-8% drop in AF. The representation decoupler is designed to capture the semantic information that determines the labels, reducing the model's reliance on domain-specific information influenced by dynamic distribution shifts. This improves the model's ability to correctly classify new classes from new domains by decoupling the node representations learned by the GNN, thereby enhancing adaptability to different domains. Additionally, minimizing the learning of irrelevant environmental features improves prediction performance. (2) In DINGLE w/o Teacher, we remove the teacher-student knowledge distillation module and train only the student model for each task. We observe a 9%-14% drop in AA and a 7%-16% drop in AF. This result highlights the importance of maintaining semantic information between old and new tasks while minimizing the correlation of domain-specific information across tasks. Retaining semantic similarity between the teacher and student models enables the student to inherit the teacher's knowledge of old classes. By maximizing the domain difference between the teacher and student, we reduce the student's dependency on outdated domain information, preventing the misprediction of old classes caused by the integration of irrelevant domain knowledge. (3) In DINGLE w/o Weighted, we remove the parameter weighting module, such that the student model inherits parameters from the previous task. Experimental results show a 3%-10% drop in AA and a 5%-8% drop in AF. These findings suggest that weighting the semantic and variational encoder parameters from all previous tasks helps preserve both semantic and domain-specific information from old tasks, particularly those distant from the current task. This preserves crucial information for all encountered classes, improving long-term performance.

Sensitive analysis. The number of node representations stored in the Representative Node Feature (RNF) Bank for each class, denoted as m, is a key hyperparameter. We investigate its impact across all datasets by varying $m \in \{1, 10, 20, 30, 40, 50\}$, and present

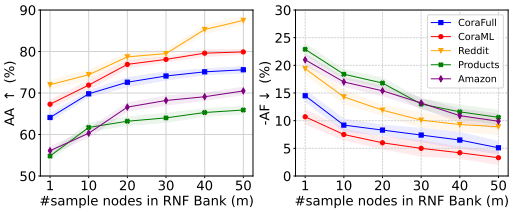

Figure 7: Analysis of the number m of selected representative node features in RNF Bank on all datasets.

the results for AA and AF in Figure 7. As the number of node representations in the RNF Bank increases, AA gradually rises (indicating improved prediction performance), while negated AF (*i.e.,* -AF) decreases (indicating reduced catastrophic forgetting). As expected, increasing the number of nodes stored enhances the model's ability to mitigate catastrophic forgetting on all datasets. The node representations in the RNF Bank retain both the attributes and topological information of experienced nodes from previous tasks, which helps preserve knowledge from previous tasks.

6 Conclusion

We introduced DINGLE, a novel framework for class-domain incremental learning on graphs, addressing covariate and label shifts simultaneously while mitigating catastrophic forgetting. By incorporating a representation decoupler and a teacher-student knowledge distillation module, DINGLE effectively disentangles node representations and preserves past knowledge through memory replay. Experimental results on 5 real-world datasets demonstrate its superiority over 11 state-of-the-art baselines, achieving higher classification accuracy and better knowledge retention.

7 Acknowledgments

This work done by Qin Tian, Minglai Shao, and Wenjun Wang is supported by the National Natural Science Foundation of China (NO.62472305), the Key R&D projet in Hainan province, China (NO.ZDYF2024SHFZ051), and Hainan Tropical Ocean Institute Yazhou Bay Innovation Research Institute Major Science and Technology Plan Project (NO.2023CXYZD001). Chen Zhao, Xintao Wu, Xujiang Zhao, and Dong Li did not receive any financial support for this work and contributed only by developing the research ideas, participating in discussions, and providing feedback on the manuscript.

References

[1] Aleksandar Bojchevski and Stephan Günnemann. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815* (2017).

[2] Antonio Carta, Andrea Cossu, Federico Errica, and Davide Bacciu. 2022. Catastrophic forgetting in deep graph networks: A graph classification benchmark. *Frontiers in artificial intelligence* 5 (2022), 824655.

[3] Sihao Ding, Fuli Feng, Xiangnan He, Yong Liao, Jun Shi, and Yongdong Zhang. 2022. Causal incremental graph convolution for recommender system retraining. *IEEE Transactions on Neural Networks and Learning Systems* 35, 4 (2022), 4718–4728.

[4] Falih Gozi Febrinanto, Feng Xia, Kristen Moore, Chandra Thapa, and Charu Aggarwal. 2023. Graph lifelong learning: A survey. *IEEE Computational Intelligence Magazine* 18, 1 (2023), 32–51.

[5] Lukas Galke, Benedikt Franke, Tobias Zielke, and Ansgar Scherp. 2021. Lifelong Learning of Graph Neural Networks for Open-World Node Classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*. 1–8. doi:10.1109/IJCNN52387.2021.9533412

[6] Lukas Galke, Iacopo Vagliano, Benedikt Franke, Tobias Zielke, Marcel Hoffmann, and Ansgar Scherp. 2023. Lifelong learning on evolving graphs under the constraints of imbalanced classes and new classes. *Neural Networks* 164 (2023), 156–176.

[7] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. 2022. Good: A graph out-of-distribution benchmark. *Advances in Neural Information Processing Systems* 35 (2022), 2059–2073.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[9] Thanh Duc Hoang, Duy-Hung Nguyen, Bao-Sinh Nguyen, Huy Hoang Nguyen, Hung Le, et al. 2023. Universal Graph Continual Learning. *Transactions on Machine Learning Research* (2023).

[10] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[11] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. 2018. Multimodal unsupervised image-to-image translation. In *Proceedings of the European conference on computer vision (ECCV)*. 172–189.

[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[14] Xiaoyu Kou, Yankai Lin, Shaobo Liu, Peng Li, Jie Zhou, and Yan Zhang. 2020. Disentangle-based continual graph representation learning. *arXiv preprint arXiv:2010.02565* (2020).

[15] Jialu Li, Yu Wang, Pengfei Zhu, Wanyu Lin, and Qinghua Hu. [n. d.]. What Matters in Graph Class Incremental Learning? An Information Preservation Perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[16] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[17] Huihui Liu, Yiding Yang, and Xinchao Wang. 2021. Overcoming catastrophic forgetting in graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 8653–8661.

[18] Jiajun Liu, Wenjun Ke, Peng Wang, Ziyu Shang, Jinhua Gao, Guozheng Li, Ke Ji, and Yanhe Liu. 2024. Towards continual knowledge graph embedding via incremental distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 8759–8768.

[19] Shikun Liu, Tianchun Li, Yongbin Feng, Nhan Tran, Han Zhao, Qiang Qiu, and Pan Li. 2023. Structural re-weighting improves graph domain adaptation. In *International conference on machine learning*. PMLR, 21778–21793.

[20] Yang Liu, Xiang Ao, Fuli Feng, Yunshan Ma, Kuan Li, Tat-Seng Chua, and Qing He. 2023. FLOOD: A flexible invariant learning framework for out-of-distribution generalization on graphs. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 1548–1558.

[21] Gianfranco Lombardo, Agostino Poggi, and Michele Tomaiuolo. 2022. Continual representation learning for node classification in power-law graphs. *Future Generation Computer Systems* 128 (2022), 420–428.

[22] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[23] Bin Lu, Xiaoying Gan, Lina Yang, Weinan Zhang, Luoyi Fu, and Xinbing Wang. 2022. Geometer: Graph few-shot class-incremental learning via prototype representation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 1152–1161.

[24] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3 (2000), 127–163.

[25] Mehrnoosh Mirtaheri, Mohammad Rostami, and Aram Galstyan. 2023. History repeats: Overcoming catastrophic forgetting for event-centric temporal knowledge graph completion. *arXiv preprint arXiv:2305.18675* (2023).

[26] Massimo Perini, Giorgia Ramponi, Paris Carbone, and Vasiliki Kalavri. 2022. Learning on streaming graphs with experience replay. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 470–478.

[27] Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference* 90, 2 (2000), 227–244.

[28] Junwei Su, Difan Zou, Zijun Zhang, and Chuan Wu. 2023. Towards robust graph incremental learning on evolving graphs. In *International Conference on Machine Learning*. PMLR, 32728–32748.

[29] Zhen Tan, Kaize Ding, Ruocheng Guo, and Huan Liu. 2022. Graph few-shot class-incremental learning. In *Proceedings of the fifteenth ACM international conference on web search and data mining*. 987–996.

[30] Qin Tian, Chen Zhao, Minglai Shao, Wenjun Wang, Yujie Lin, and Dong Li. 2025. Mldgg: Meta-learning for domain generalization on graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*. 1361–1372.

[31] Zonggui Tian, Du Zhang, and Hong-Ning Dai. 2024. Continual Learning on Graphs: A Survey. *arXiv preprint arXiv:2402.06330* (2024).

[32] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[33] Chen Wang, Yuheng Qiu, Dasong Gao, and Sebastian Scherer. 2022. Lifelong graph learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 13719–13728.

[34] Junshan Wang, Wenhao Zhu, Guojie Song, and Liang Wang. 2022. Streaming graph neural networks with generative replay. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1878–1888.

[35] Ke Wang, Senqiang Zhou, Chee Ada Fu, and Jeffrey Xu Yu. 2003. Mining changes of classification by correspondence tracing. In *Proceedings of the 2003 SIAM International Conference on Data Mining*. SIAM, 95–106.

[36] Yuening Wang, Yingxue Zhang, and Mark Coates. 2021. Graph structure aware contrastive knowledge distillation for incremental learning in recommender systems. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3518–3522.

[37] Hou Yifan, Zhang Jian, Cheng James, Ma Kaili, Ma Richard TB, Chen Hongzhi, and Yang Ming-Chang. 2020. Measuring and improving the use of graph information in graph neural network. In *The Eighth International Conference on Learning Representations (ICLR 2020), Addis Ababa*.

[38] XU Yishi, Yingxue Zhang, GUO Huifeng, Ruiming Tang, and GENG Yanhui. 2023. Graph structure aware incremental learning for recommender system. US Patent App. 18/111,066.

[39] Qiao Yuan, Sheng-Uei Guan, Pin Ni, Tianlun Luo, Ka Lok Man, Prudence Wong, and Victor Chang. 2023. Continual graph learning: A survey. *arXiv preprint arXiv:2301.12230* (2023).

[40] Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. 2023. Continual learning on dynamic graphs via parameter isolation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 601–611.

[41] Xikun Zhang, Dongjin Song, Yixin Chen, and Dacheng Tao. 2024. Topology-aware embedding memory for continual learning on expanding networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4326–4337.

[42] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Cglb: Benchmark tasks for continual graph learning. *Advances in Neural Information Processing Systems* 35 (2022), 13006–13021.

[43] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2022. Sparsified subgraph memory for continual graph representation learning. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1335–1340.

[44] Xikun Zhang, Dongjin Song, and Dacheng Tao. 2024. Continual Learning on Graphs: Challenges, Solutions, and Opportunities. *arXiv preprint arXiv:2402.11565* (2024).

[45] Fan Zhou and Chengtai Cao. 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 4714–4722.

A Notations

For clear interpretation, we list the notations used in this paper and their corresponding explanation, as shown in Table 3.

Table 3: Important notations and corresponding descriptions.

Notations	Descriptions
$\mathcal{G}$	A set of graphs
$\mathcal{V}$	A set of nodes of a graph
G_t	A graph of task t
A_t	The adjacency matrix of G_t
X_t	The node feature matrix of G_t
$\mathbf{y}_t$	Node labels of G_t
e_t	A domain indicator specific to G_t
C_t	The set of classes at task t
ΔC_t	The set of incremented classes
T	Total number of tasks
E^c	Semantic encoder
E^v	Variation encoder
D	Decoder
h	Node classification head
$\hat{\mathbf{v}}$	A variation factor that is randomly sampled from the prior distribution $\mathcal{N}(0, \mathbf{I})$
$\mathbf{c}$	Semantic factor
$\mathbf{v}$	Variation factor
Φ	Discriminator
Z	Node representations learned from GNNs
$\boldsymbol{\theta}^c$	Parameters of the semantic encoder
$\boldsymbol{\theta}^v$	Parameters of the variation encoder
m	The number of representative node features stored in the RNF bank
$\bar{\mathbf{z}}_{k,t}$	The prototype of the node representations for the k-th class at task t

B Datasets Setting

We summarize the construction of tasks for each dataset as follows:

- CoraFull[1] is a paper citation network with 70 paper topics, where nodes represent papers and edges denote citation relationships. The dataset is partitioned into 11 domains based on the word of papers following [7]. Task 0 includes 20 base classes from one domain. Each of the next 10 tasks introduces 5 new classes from a different domain.
- Similarly, CoraML[1] consists of 7 paper topics divided into 6 domains, with 2 classes as base classes and each task incremented by 1 class from different domains.
- Reddit[8] is an online discussion forum with 40 communities of posts, where nodes represent posts and edges denote interactions between them, with an undirected edge created if two posts are interacted with by the same user in the comments section. The dataset is partitioned into 11 domains based on node degree, with 20 classes from one domain as base classes and 2 classes from each of the remaining 10 domains as incremental additions.
- Products[10] is a co-purchase network with 47 product categories, where nodes represent goods and edges indicate frequently co-purchased items. The dataset is divided into 11 domains based on node degree, with 27 classes as base classes and an increment of 2 classes from distinct domains.
- Amazon [37] is a co-purchase network of computer-related products, where nodes represent goods and edges indicate frequently co-purchased items. Similar to Products, this dataset is divided into 9 domains based on node degree, with 2 classes selected as base classes and 1 class added incrementally for each task.

C Baselines

- PDGNNs [41] preserves topology information based on memory replay techniques while reducing the memory space complexity.
- GSIP [15] preserves the information of old models to calibrate node semantic and graph structure shifts.
- Geometer [23] introduces a graph few-shot class-incremental learning framework by learning the prototype of each class.
- FGN [33] converts the graph data into independent data with the feature map to apply traditional continuous learning methods.
- LLGNN [5] introduces a new metric based on k- neighborhood time differences to balance the influence of implicit and explicit knowledge on historical data changes for lifelong learning on graphs.
- SSRM [28] mitigates the impact of the structural shift on catastrophic forgetting based on regularization.
- SSM [43] stores sampled sparse subgraphs in memory buffer to preserve topology information of old tasks.
- ER-GNN [45] replays the node embedding of previous tasks when learning new tasks.
- TWP [17] preserves the parameters playing pivotal roles in the topological aggregation to overcome catastrophic forgetting.
- EWC [13] imposes a quadratic penalty on the model weights based on their relevance to previous tasks.
- LWF [16] utilizes information distillation to reduce the discrepancy between old and new models.

D Experiment Results

Efficiency analysis of DINGLE. We evaluate the runtime of DINGLE and all baselines all tasks on the five datasets, with results presented in Figure 8. The efficiency results show that DINGLE achieves classification improvements in class-incremental and domain-incremental scenarios while maintaining a runtime comparable to state-of-the-art baselines, demonstrating its effectiveness without introducing significant computational overhead.

The effectiveness of DINGLE using different backbones. Additionally, we evaluated the node classification performance of all baseline methods using different GNN backbones (GAT [32] and GraphSAGE [8]) on the CoraFull dataset, with results presented in Table 4 and Figure 9. DINGLE maintains strong performance across backbones. Particularly, attention-based methods like Geometer

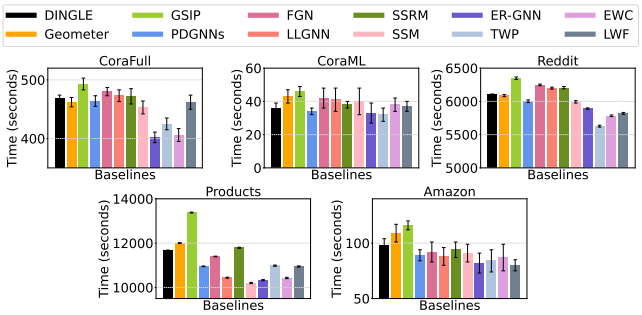

Figure 8: Efficiency comparison of DINGLE and all baselines on all datasets.

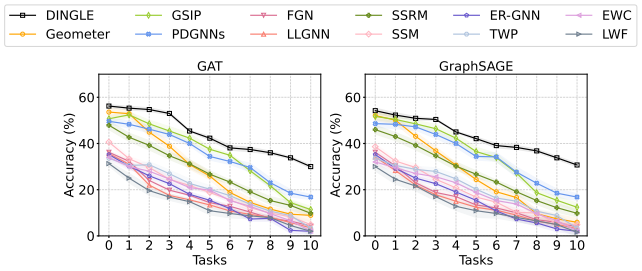

Figure 9: Performance comparisons for each task using GAT and GraphSAGE backbones on the CoraFull dataset.

perform slightly better with GAT, while backbone changes have a relatively minor impact on other methods.

Table 4: Overall performance comparisons using GAT and GraphSAGE backbones on the CoraFull dataset.

Methods	GAT [32]		GraphSAGE [8]	
	AA/% ↑	AF/% ↑	AA/% ↑	AF/% ↑
LWF [16]	35.8 ± 2.9	-43.2 ± 3.6	36.0 ± 4.8	-43.6 ± 3.6
EWC [13]	42.2 ± 3.8	-40.1 ± 3.5	42.0 ± 2.8	-40.8 ± 3.6
TWP [17]	41.0 ± 2.6	-43.8 ± 2.9	39.5 ± 3.2	-44.4 ± 2.7
ER-GNN [45]	50.8 ± 3.9	-28.4 ± 3.2	50.5 ± 4.8	-24.6 ± 5.2
SSM [43]	48.6 ± 3.8	-29.9 ± 3.4	45.8 ± 3.1	-28.6± 4.9
SSRM [28]	53.5 ± 3.3	-24.0 ± 4.8	51.2 ± 4.6	-26.0 ± 3.7
LLGNN [5]	43.2 ± 3.5	-42.7 ± 4.9	41.6 ± 4.6	-45.3 ± 5.0
FGN [33]	48.9 ± 1.3	-30.4 ± 1.9	46.5 ± 2.0	-32.8 ± 1.8
PDGNNs [41]	56.5 ± 2.8	-14.5 ± 3.0	55.4 ± 2.8	-13.9 ± 3.8
GSIP [15]	52.1 ± 3.8	-20.8 ± 4.0	51.9 ± 3.2	-21.0 ± 2.8
Geometer [23]	57.9 ± 3.9	-18.2 ± 3.5	56.0 ± 2.9	-22.2 ± 3.0
DINGLE (Ours)	**62.8 ± 1.6**	**-5.3 ± 2.0**	**61.2 ± 1.4**	**-5.2 ± 1.8**