UFO: A Unified Flow-Oriented Framework for Robust Continual Graph Learning

Danhui Zhang[1], **Zhe Wang**[1], **Qing Qing**[1], **Jiarui Liu**[1], **Wentao Gao**[2], **Ziqi Xu**[3], **Mingliang Hou**[4], **Xikun Zhang**[3*], **Renqiang Luo**[1*]
[1]Jilin University, [2]Adelaide University, [3]RMIT University, [4]Jinan University

Abstract

Graph learning research has increasingly shifted toward continual graph learning (CGL), which better reflects real-world scenarios where graphs evolve over time. However, existing CGL methods largely assume clean supervision and overlook a critical challenge: the newly arriving portions of the graph are often noisy, due to annotation errors or adversarial corruption. This mismatch limits their applicability in practice. In this work, we study robust continual graph learning, where models must simultaneously handle catastrophic forgetting and noisy supervision in evolving graph data. We present the first investigation of a failure mode in this setting—catastrophic remembering, where models persistently reinforce corrupted knowledge across tasks. To address these challenges, we propose a Unified Flow-Oriented framework (UFO). First, UFO models conditional feature distributions via flow-based generative modeling and produces replay representations, mitigating forgetting without storing historical data. Second, UFO estimates instance-level reliability scores to distinguish clean from noisy nodes, reducing the impact of corrupted supervision and alleviating catastrophic remembering. Extensive experiments on four benchmark graph datasets under varying noise ratios demonstrate that UFO consistently outperforms existing methods in both accuracy and forgetting metrics. Code is available at: `https://github.com/Zhang-danhui/UFO.git`.

1 Introduction

Graphs in real-world applications [1, 2, 3] are inherently dynamic and evolve over time [4]. In social, citation, and co-authorship networks, new entities and connections emerge continuously [5, 6]. To handle this data stream, models must acquire new knowledge while preserving previously learned information. This necessity has established Continual Graph Learning (CGL) as a critical research area [7], focusing on learning from sequential graph data.

Despite its potential, CGL faces two primary obstacles. First, storage limitations and privacy concerns [8, 9] often make historical data inaccessible, leading to catastrophic forgetting [10]. Second, while most CGL models assume clean training labels, label noise is inevitable in practice. Real-world datasets often contain significant noise ratios, sometimes exceeding 30% [11]. On graphs, this noise is particularly destructive in that it is not forgotten but retained across tasks. As illustrated in Table 1, more mislabeled nodes enter the high-confidence region in later tasks, as the message-passing mechanism

Table 1: Number of mislabeled nodes in the high-confidence region on CS using GCN. The first (second) row corresponds to a 15% (30%) noise ratio.

Confidence \ Task	1	2	3	4	5
Top 15%	8	41	24	141	26
Top 30%	21	62	130	93	33

*Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

propagates and amplifies errors across neighborhoods [12]. This forces the model to memorize incorrect information, causing catastrophic remembering [13], a concept we extend to CGL.

Current techniques rarely address both challenges simultaneously. Standard CGL methods rely on regularization or replay buffers to prevent forgetting [14], yet they neglect the quality of labels and fail to handle noise. Conversely, robust learning methods use sample selection or robust loss functions to mitigate noise [15], but they typically focus on the current task and ignore historical knowledge. Therefore, their effectiveness is strongly tied to accessible data, leaving knowledge learned from inaccessible data vulnerable to forgetting [16]. While a few hybrid approaches [13, 17, 18] have been proposed for image data, they cannot perform as well on graphs. This is because noise propagates through the message-passing mechanism. The complex topology of graphs further complicates the distinction between noise and legitimate structural shifts [19].

To bridge this gap, we propose a unified perspective in which catastrophic forgetting and noise robustness can be addressed simultaneously. Our key insight is that both challenges can be mitigated by learning and leveraging the underlying feature distributions of the data. By aligning new observations with these distributions, the model can suppress noisy supervision, while the distributions themselves enable knowledge retention without storing historical data.

Concretely, we replace explicit data storage with a generative distribution model. This model synthesizes representative features of past tasks for replay, preserving performance while avoiding the need to retain sensitive or large-scale historical data. At the same time, we estimate the reliability of incoming nodes based on their consistency with the learned distributions, allowing the model to down-weight noisy or corrupted samples during training.

Building on this idea, we propose **UFO**, a unified framework for robust continual graph learning without historical data storage. UFO models conditional feature distributions to generate replay representations, effectively mitigating catastrophic forgetting. To handle label noise, UFO incorporates an instance-level adaptation mechanism that assigns importance weights to nodes based on their alignment with the learned feature space, ensuring that reliable samples dominate model updates. In addition, a knowledge preservation strategy constrains representation drift across tasks, maintaining both structural and semantic consistency over time.

In summary, our main contributions are as follows:

- We introduce the problem of robust continual graph learning with label noise, and present the first investigation of a failure mode in CGL, catastrophic remembering, caused by the accumulation of corrupted supervision.
- We propose UFO, a unified framework that combines distribution-based feature replay and instance-level reliability estimation to address forgetting and noisy labels without storing historical data.
- We conduct extensive experiments on multiple benchmark datasets, demonstrating that UFO consistently improves both predictive performance and robustness under varying noise levels.

2 Related Work

2.1 Continual Graph Learning

Continual graph learning [20, 21] is a paradigm that enables models to learn from evolving graphs while mitigating catastrophic forgetting. Existing approaches are divided into three families: (1) Regularization-based methods introduce a penalty term in the loss function to constrain the updates of crucial parameters. Early studies like LwF [22] learn the soft labels from the old model and TWP [23] preserves the topological aggregation of previous graphs. (2) Experience replay-based methods store or generate representative samples from previous tasks, e.g., DSLR [24] introduces coverage-based diversity to select informative nodes. Since generative models can learn intricate data distributions and produce samples [25, 26], they can also be used for replay. DMSG [27] takes into account memory diversity through selection and uses a variational layer to synthesize node embeddings. (3) Parameter isolation-based methods use different parameters for different tasks, so that learning a new task has less effect on old tasks. HPNs [14] dynamically learns different levels of prototypes. However, all these methods are dependent on clean supervision, which limits their practicality in real-world scenarios [28]. In this work, we consider a more challenging problem in

robust continual graph learning: how to learn from sequential tasks under corrupted supervision when historical data are inaccessible? This requires the model to preserve correct knowledge from seen tasks while avoiding the reinforcement of corrupted information.

2.2 Graph Learning with Label Noise

Learning with label noise [29] focuses on training models with corrupted data. There have been many efforts to enhance robustness, such as robust loss design [30], label correction [31], and sample selection [32]. For example, SCE [33] improves robustness by combining cross entropy with reverse cross entropy. Bootstrap [34] replaces the target labels with a combination of their predicted labels. Recent works have extended these ideas to graph learning. CLNode [35] proposes a curriculum learning framework for node classification with a multi-perspective difficulty measurer. TFR [36] leverages a backbone GNN and a decoder GNN to reconstruct the feature distribution to maximize mutual information. TSS [37] utilizes graph topology to progressively select confident nodes and trains the model from easy to hard. However, existing methods focus on the current task and overlook real-world dynamic graph environments, where historical graph data may be unavailable due to privacy or storage constraints, making previously learned knowledge vulnerable to forgetting.

3 Problem Statement

3.1 Notations

Consider a sequence of n tasks $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_t, \cdots, \mathcal{T}_n\}$. Each task $\mathcal{T}_t$ is defined on a graph $\mathcal{G}_t = \{\mathcal{V}_t, \mathcal{E}_t\}$, where $\mathcal{V}_t$ is the node set and $\mathcal{E}_t$ is the edge set. The graph structure is represented by the adjacency matrix $A_t \in \mathbb{R}^{N_t \times N_t}$, where $N_t = |\mathcal{V}_t|$. Each node $v_i \in \mathcal{V}_t$ has an input feature $h_i^{(0)}$ and an observed label y_i. During the training stage of task $\mathcal{T}_t$, the node set $\mathcal{V}_t$ is divided into a training set $\mathcal{V}_t^{tr}$, a validation set $\mathcal{V}_t^{val}$, and a test set $\mathcal{V}_t^{te}$. The observed labels may be noisy in $\mathcal{V}_t^{tr}$ and clean in $\mathcal{V}_t^{val}$ and $\mathcal{V}_t^{te}$. Formally, our objective is to learn sequential tasks under corrupted supervision while preserving previously learned knowledge: $\min_{\Theta_t} \frac{1}{t} \sum_{j=1}^{t} \mathcal{L}_j(\Theta_t)$, where Θ_t refers to the model parameters for the t-th task.

3.2 Preliminary

Given an input $x \in \mathcal{X}$ and a latent variable $z \in \mathcal{Z}$, a normalizing flow [38, 39] defines an invertible mapping $f : \mathcal{X} \to \mathcal{Z}$ that transforms the original distribution into a base distribution, such as a standard Gaussian distribution. Here, $z = f(x)$ and $x = f^{-1}(z)$. By the change of variables formula, the likelihood of x can be written as:

$$p_X(x) = p_Z(z) \left| \det \frac{\partial f(x)}{\partial x} \right|, \tag{1}$$

where $\left| \det \frac{\partial f(x)}{\partial x} \right|$ is the absolute Jacobian determinant of the transformation f. In addition, let $y \in \mathcal{Y}$ be the class label, and the flow can be extended to the conditional setting [40]. The invertible mapping becomes $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, where $z = f(x, y)$ and $x = f^{-1}(z, y)$. We can express the conditional density of x by:

$$p_{X|Y}(x \mid y) = p_Z(z) \left| \det \frac{\partial f(x, y)}{\partial x} \right|. \tag{2}$$

In general, a single transformation is not flexible enough to model a complex distribution [41]. Therefore, f is defined as a composition of K simple invertible transformations: $f = f_K \circ \cdots \circ f_1$. The $\circ$ denotes function composition. Based on Eq. (2), the conditional log-likelihood of x under label y is:

$$\log p_{X|Y}(x \mid y) = \log p_Z(z) + \sum_{k=1}^{K} \log \left| \det \frac{\partial z_k}{\partial z_{k-1}} \right|, \tag{3}$$

where z_k is the intermediate variable after the k-th transformation.

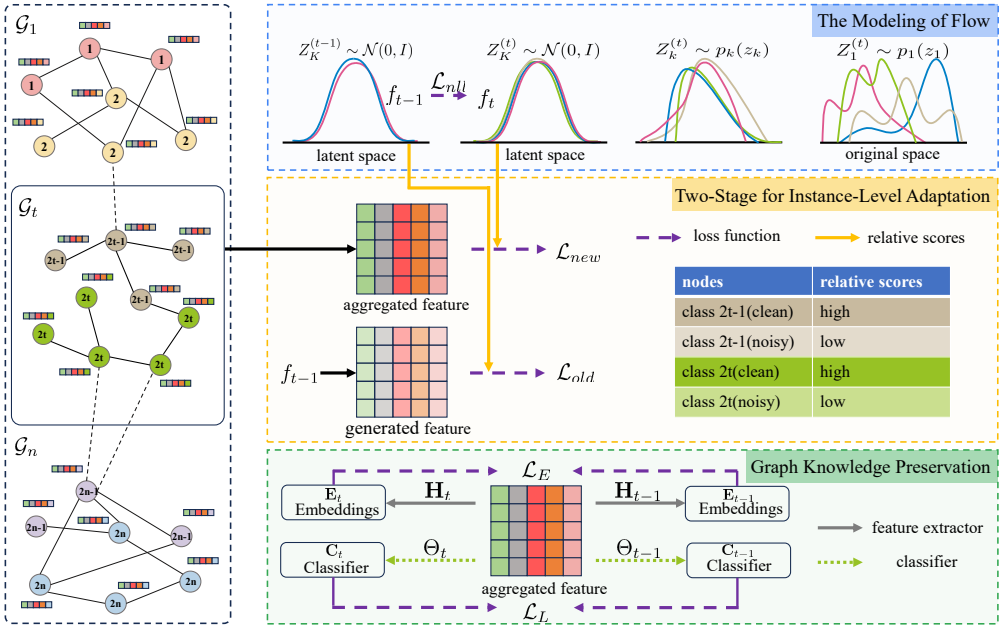


Figure 1: The framework of UFO: (1) The flow models feature distributions over sequential graphs and generates features for previous tasks. (2) Relative scores are computed using f_t and f_{t-1}, and are used to modulate $\mathcal{L}_{\text{new}}$ and $\mathcal{L}_{\text{old}}$ for instance-level adaptation. (3) Knowledge preservation constraints maintain structural and semantic consistency across tasks.

4 Methodology

This section introduces UFO, as illustrated in Figure 1. The following sections describe three related modules that work together. First, Section 4.1 introduces the modeling of flow, which maps feature distributions into a shared standard Gaussian latent space and uses the learned distributions to generate replay representations for earlier tasks. Second, Section 4.2 presents the instance-level adaptation mechanism. At task $\mathcal{T}_t$, the current flow f_t assigns dynamic scores to instances according to their alignment with the task distribution, reducing the effect of noisy supervision in $\mathcal{L}_{\text{new}}$. The frozen flow f_{t-1} scores generated replay features in the historical feature space, down-weighting low-likelihood samples in $\mathcal{L}_{\text{old}}$. Finally, Section 4.3 introduces graph knowledge preservation, which constrains structural and semantic consistency across tasks.

4.1 The Modeling of Flow

Flow is a type of powerful generative model [39]. In UFO, it plays a dual role: synthesizing representative features of prior tasks and evaluating the tractable class-conditional likelihood. Unlike VAEs [42] and diffusion models [43], which approximate the likelihood with the evidence lower bound (ELBO) or costly numerical estimation, flow computes it exactly via the change-of-variables formula. To enable such a model to learn informative graph representations, we employ the graph neural network (GNN) proposed by [44, 45]. Given the neighborhood set $\mathcal{N}(v)$ of node v, the embedding $h_v^{(l)}$ at the l-th layer is calculated as follows:

$$h_{\mathcal{N}(v)}^{(l)} = \text{AGGREGATE}^{(l)}\left(\left\{h_u^{(l-1)},\ \forall u \in \mathcal{N}(v)\right\}\right), \tag{4}$$

$$h_v^{(l)} = \text{COMBINE}^{(l)}\left(h_v^{(l-1)}, h_{\mathcal{N}(v)}^{(l)}\right). \tag{5}$$

Eq. (5) shows that the embedding of node v is updated by combining its own representation with the aggregated information from its neighbors. Following this message-passing mechanism, we construct the aggregated feature $x = h_{\mathcal{N}(v)}^{(l)}$. Then, the flow models the conditional distribution $p_{X|Y}(x \mid y)$ for each class to capture the correlation between features and labels. Specifically, it adopts a shared

standard Gaussian prior $\mathcal{N}(0, I)$ in the latent space and uses y as conditional information. The negative log-likelihood at task $\mathcal{T}_t$ is formulated as:

$$\mathcal{L}_{nll} = -\sum_{v_i \in \mathcal{V}_t^{tr}} \log p_{X|Y}(x_i \mid y_i), \tag{6}$$

where $\mathcal{V}_t^{tr}$ is the training set of task $\mathcal{T}_t$. However, directly training the flow model on a single task leads to a bias toward the current distribution. Inspired by [46], we incorporate a replay strategy to mitigate this issue, optimizing the flow with the current and generated features:

$$\mathcal{L}_{nll} = -\sum_{v_i \in \mathcal{V}_t^{tr}} \log p_{X|Y}(x_i \mid y_i) - \sum_{(\tilde{z}_i, \tilde{y}_i) \in \bigcup_{j=1}^{t-1} \tilde{B}_j} \log p_{X|Y}(f_{t-1}^{-1}(\tilde{z}_i, \tilde{y}_i) \mid \tilde{y}_i), \tag{7}$$

where $\tilde{B}_j = \{(\tilde{z}_i, \tilde{y}_i)\}_{i=1}^{b}$ is the set of size b containing sampled latent variables $\tilde{z}_i$ with labels $\tilde{y}_i$ from historical task $\mathcal{T}_j$. f_{t-1}^{-1} synthesizes replay features by mapping these variables to the original space, helping the flow model remember what it has learned in the past.

4.2 Two-Stage for Instance-Level Adaptation

In continual learning with label noise, corrupted labels can degrade the performance of GNNs and disrupt the model's inherent stability, as noisy information may spread through graph structure [47, 48]. This effect can accumulate across tasks and cause catastrophic remembering. To resolve the above problem, we propose two-stage training to adaptively mitigate the negative effects. In Stage 1, we estimate the importance of each instance with conditional log-likelihood $r_i = \log p_{X|Y}(x_i \mid y_i)$. Since softmax is shift-invariant, i.e., $\mathrm{softmax}(\mathbf{r}) = \mathrm{softmax}(\mathbf{r} - m(\mathbf{r}))$, the relative scores $\mathbf{s}$ are computed as:

$$m(\mathbf{r}) := \max_i r_i, \qquad \mathbf{s} = b \cdot \mathrm{softmax}\left(\mathbf{r} - m(\mathbf{r})\right), \tag{8}$$

where b is the batch size of the current task. Additionally, during learning a new task, the scores are smoothed to reduce outliers during early flow training and clipped to the interval $[\alpha, \beta]$: $s_i = \min\left(\beta, \max\left(\alpha, s_i\right)\right)$. Let s_i and $\tilde{s}_i$ denote the scores computed by the current flow f_t and frozen flow f_{t-1}. In Stage 2, we leverage the scores to train the model through two objectives: $\mathcal{L}_{ce}^{new}$ and $\mathcal{L}_{ce}^{old}$, which are defined as weighted cross-entropy losses. Specifically, $\mathcal{L}_{ce}^{new}$ regulates the contribution of each incoming node (e.g., clean nodes exert a stronger influence), preventing the graph model from being misled by noisy nodes. Without this term, all incoming nodes would be treated as equally reliable. The loss is formulated as:

$$\mathcal{L}_{ce}^{new} = \sum_{v_i \in \mathcal{V}_t^{tr}} s_i \ell_{ce}\left(C_t(H_t(v_i)), y_i\right), \tag{9}$$

where $\ell_{ce}(\cdot)$ refers to the cross-entropy loss, and H_t is the GNN encoder at task $\mathcal{T}_t$, followed by a classifier C_t. For previous tasks, the second objective is targeted at historical replay features and helps prevent the reinforcement of erroneous knowledge:

$$\mathcal{L}_{ce}^{old} = \sum_{(\tilde{z}_i, \tilde{y}_i) \in \bigcup_{j=1}^{t-1} \tilde{B}_j} \tilde{s}_i \ell_{ce}\left(C_t\left(H_t\left(f_{t-1}^{-1}(\tilde{z}_i, \tilde{y}_i)\right)\right), \tilde{y}_i\right). \tag{10}$$

4.3 Graph Knowledge Preservation

As a sequence of graphs arrives, the original feature space gradually shifts, which alters the representations learned from previous tasks and weakens the stability of the latent space [22]. Since the flow relies on a stable feature space to model distributions, we preserve graph knowledge to control the shifts and maintain the learned structural and semantic information [49]. The embeddings are used to anchor the representation of the same node v_i:

$$\mathcal{L}_E = \sum_{v_i \in \mathcal{V}_t^{tr}} \left\| H_t(v_i) - H_{t-1}(v_i) \right\|_2^2, \tag{11}$$

where H_{t-1} is the frozen GNN encoder at task $\mathcal{T}_{t-1}$. Class semantic knowledge is encoded in the output distribution of the classifier, which reflects the relations among classes. Directly optimizing the

Algorithm 1: Framework of UFO

Input: A sequence of n graphs $\{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_n\}$, flow f, graph encoder H, classifier C
Output: Updated H_n and C_n
for $t \leftarrow 1$ **to** n **do**
  if $t = 1$ **then**
    Aggregate features $\mathbf{x}_1$ from $\mathcal{G}_1$;
    Train f_1 on $\mathbf{x}_1$ in Eq. (6);
    Compute the relative scores $\mathbf{s}_1$ from f_1 in Eq. (8);
    Train H_1 and C_1 on $\mathcal{G}_1$ in Eq. (9);
  else
    Aggregate features $\mathbf{x}_t$ from $\mathcal{G}_t$;
    Generate features $\tilde{\mathbf{x}}_{t-1}$ from f_{t-1};
    Train f_t on $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_{t-1}$ in Eq. (7);
    Compute the relative scores $\mathbf{s}_t$ and $\tilde{\mathbf{s}}_{t-1}$ from f_t and f_{t-1} in Eq. (8);
    Train H_t and C_t on $\mathcal{G}_t$ and $\tilde{\mathbf{x}}_{t-1}$ in Eq. (14);
  end
end

current classifier may disrupt this previously learned knowledge. We therefore use the soft predictions from the frozen classifier C_{t-1} to guide the current classifier C_t. The two classifiers are evaluated with the same task head, and the Kullback–Leibler (KL) divergence is used for distillation:

$$\mathcal{L}_L = \sum_{v_i \in \mathcal{V}_t^{tr}} \tau^2 \mathrm{KL}\left(\mathrm{softmax}(C_{t-1}(H_{t-1}(v_i))/\tau) \parallel \mathrm{softmax}(C_t(H_t(v_i))/\tau)\right), \tag{12}$$

where τ is a temperature parameter, and τ^2 is used to compensate for the $1/\tau^2$ scaling of the gradients. By optimizing the above two losses, structural and semantic consistency is maintained jointly:

$$\mathcal{L}_{KP} = \alpha_E \mathcal{L}_E + \alpha_L \mathcal{L}_L, \tag{13}$$

where α_E and α_L are loss weights. Finally, the learning objective for the GNN is defined as the sum of the losses:

$$\mathcal{L}_{graph} = \mathcal{L}_{ce}^{new} + \mathcal{L}_{ce}^{old} + \mathcal{L}_{KP}. \tag{14}$$

And algorithm 1 summarizes the whole training process of UFO.

5 Experiments

5.1 Experimental Setup

Datasets and Implementation Details. We evaluate our method on four real-world datasets: CoraFull [5], CS [50], WikiCS [51], and Photo [50]. More detailed statistics are provided in Appendix A.1. Following [20], we divide the classes into a sequence of tasks for each dataset, and split the nodes in each task into 60%, 20%, and 20% for training, validation, and testing, respectively. We add symmetric noise to a fraction of labels in the training set while keeping the validation and test sets clean. For each selected node, its label is replaced by a label uniformly sampled from the other classes within the same task. We also report the results under pair flipping label noise in Appendix B.1, where each selected label is flipped to a predefined paired class within the same task. The implementation and detailed settings are provided in Appendix A.4.

Baselines. Since there is no existing method specifically designed for our problem, we summarize eight baselines into two categories. The first category includes well-known continual learning methods, such as LwF [22], ERGNN [52], DSLR [24], and DMSG [27]. To adapt these methods to noisy labels, we introduce the SCE [33] loss during training. The second category consists of methods designed for noisy label learning, including SCE, CLNode [35], TSS [37], and TFR [36]. Since these methods are not originally designed for continual learning, we adopt the CM strategy from ERGNN

Table 2: Average performance comparison on each dataset under noise ratios of $\{0\%, 15\%, 30\%\}$. The best performance is in **red**, and the second best is in blue.

Models	Accuracy↑ 0%	Accuracy↑ 15%	Accuracy↑ 30%	Forgetting↑ 0%	Forgetting↑ 15%	Forgetting↑ 30%
CoraFull						
Joint	$90.40_{\pm 0.10}$	$75.82_{\pm 0.12}$	$62.45_{\pm 0.27}$		-	
Bare model	$34.21_{\pm 1.69}$	$31.32_{\pm 6.43}$	$30.60_{\pm 1.13}$	$-61.91_{\pm 3.50}$	$-49.15_{\pm 9.88}$	$-36.00_{\pm 1.09}$
LwF	$60.32_{\pm 0.77}$	$55.25_{\pm 2.14}$	$50.87_{\pm 1.54}$	$-33.41_{\pm 0.95}$	$-24.84_{\pm 2.82}$	$-15.57_{\pm 1.43}$
ERGNN	$77.74_{\pm 0.15}$	$40.91_{\pm 0.34}$	$33.94_{\pm 0.58}$	**$-4.62_{\pm 0.37}$**	$-33.06_{\pm 0.40}$	$-32.94_{\pm 0.37}$
DSLR	$74.23_{\pm 0.90}$	$62.09_{\pm 1.62}$	$51.06_{\pm 0.99}$	$-14.47_{\pm 0.84}$	$-11.75_{\pm 1.46}$	$-8.98_{\pm 0.75}$
DMSG	$78.04_{\pm 0.56}$	$67.20_{\pm 0.82}$	$59.71_{\pm 1.30}$	$-13.20_{\pm 0.66}$	$-10.65_{\pm 0.81}$	$-7.68_{\pm 0.85}$
SCE	$39.30_{\pm 0.15}$	$41.27_{\pm 1.21}$	$34.41_{\pm 1.77}$	$-56.65_{\pm 0.28}$	$-42.06_{\pm 1.45}$	$-35.08_{\pm 2.14}$
CLNode	$78.01_{\pm 0.18}$	$30.12_{\pm 0.81}$	$27.13_{\pm 0.22}$	$-5.06_{\pm 0.31}$	$-53.95_{\pm 0.37}$	$-55.91_{\pm 0.52}$
TSS	$75.02_{\pm 0.57}$	$27.61_{\pm 1.89}$	$25.27_{\pm 1.69}$	$-9.20_{\pm 1.30}$	$-52.97_{\pm 1.22}$	$-47.19_{\pm 1.38}$
TFR	$49.08_{\pm 0.61}$	$36.08_{\pm 1.35}$	$26.89_{\pm 1.36}$	$-45.95_{\pm 0.72}$	$-45.92_{\pm 0.61}$	$-42.35_{\pm 1.09}$
UFO	**$81.78_{\pm 0.85}$**	**$80.71_{\pm 0.08}$**	**$77.77_{\pm 1.04}$**	$-6.35_{\pm 1.10}$	**$-6.33_{\pm 2.25}$**	**$-7.10_{\pm 1.27}$**
CS						
Joint	$97.00_{\pm 0.10}$	$67.43_{\pm 4.15}$	$51.06_{\pm 1.83}$		-	
Bare model	$61.22_{\pm 0.10}$	$62.83_{\pm 0.07}$	$53.04_{\pm 0.03}$	$-45.55_{\pm 0.02}$	$-28.98_{\pm 0.01}$	$-25.10_{\pm 0.21}$
LwF	$92.51_{\pm 4.10}$	$53.39_{\pm 6.40}$	$45.19_{\pm 15.55}$	$-5.49_{\pm 4.78}$	$-40.94_{\pm 8.02}$	$-32.99_{\pm 13.22}$
ERGNN	$94.40_{\pm 0.14}$	$57.82_{\pm 2.86}$	$47.73_{\pm 3.43}$	$-3.26_{\pm 0.29}$	$-20.09_{\pm 6.03}$	$-11.67_{\pm 0.85}$
DSLR	$96.47_{\pm 0.80}$	$77.67_{\pm 0.45}$	$68.10_{\pm 1.80}$	$-2.01_{\pm 0.88}$	$-12.98_{\pm 1.59}$	$-2.29_{\pm 2.39}$
DMSG	$93.87_{\pm 1.14}$	$70.46_{\pm 3.67}$	$57.55_{\pm 8.96}$	$-4.24_{\pm 1.43}$	$-6.34_{\pm 4.45}$	**$7.13_{\pm 9.74}$**
SCE	$78.90_{\pm 7.20}$	$58.42_{\pm 8.09}$	$46.00_{\pm 6.30}$	$-22.48_{\pm 8.69}$	$-33.80_{\pm 11.78}$	$-29.68_{\pm 5.30}$
CLNode	$93.84_{\pm 0.32}$	$51.80_{\pm 1.52}$	$57.89_{\pm 1.22}$	$-3.84_{\pm 0.47}$	$-55.76_{\pm 1.90}$	$-47.60_{\pm 1.36}$
TSS	$79.62_{\pm 5.16}$	$55.28_{\pm 9.70}$	$48.12_{\pm 6.16}$	$-14.29_{\pm 8.06}$	$-41.88_{\pm 12.52}$	$-36.27_{\pm 8.29}$
TFR	$94.11_{\pm 1.98}$	$28.33_{\pm 9.63}$	$28.81_{\pm 8.76}$	$-3.43_{\pm 2.58}$	$-48.21_{\pm 15.95}$	$-11.52_{\pm 6.81}$
UFO	**$97.29_{\pm 0.29}$**	**$92.21_{\pm 0.43}$**	**$81.46_{\pm 0.43}$**	**$-1.30_{\pm 0.27}$**	**$0.06_{\pm 1.34}$**	$3.14_{\pm 0.20}$
WikiCS						
Joint	$92.70_{\pm 0.40}$	$90.88_{\pm 0.40}$	$86.41_{\pm 3.01}$		-	
Bare model	$77.72_{\pm 4.51}$	$68.95_{\pm 6.56}$	$65.24_{\pm 7.98}$	$-17.67_{\pm 5.91}$	$-21.10_{\pm 3.59}$	$-15.06_{\pm 11.64}$
LwF	$88.75_{\pm 1.54}$	$74.06_{\pm 4.37}$	$62.76_{\pm 4.96}$	$-4.72_{\pm 2.09}$	$-20.50_{\pm 6.35}$	$-20.08_{\pm 6.83}$
ERGNN	$86.58_{\pm 0.36}$	$52.77_{\pm 3.69}$	$48.62_{\pm 6.77}$	$-6.54_{\pm 0.37}$	$-44.23_{\pm 10.28}$	$-39.28_{\pm 2.36}$
DSLR	$82.71_{\pm 0.71}$	$70.35_{\pm 2.87}$	$61.74_{\pm 1.39}$	$-8.65_{\pm 0.44}$	$-12.93_{\pm 2.34}$	$-8.49_{\pm 2.93}$
DMSG	$83.69_{\pm 1.55}$	$82.18_{\pm 2.34}$	$66.11_{\pm 5.12}$	$-10.84_{\pm 1.48}$	$-9.69_{\pm 2.01}$	$-18.36_{\pm 10.72}$
SCE	$76.26_{\pm 0.40}$	$65.59_{\pm 2.69}$	$52.77_{\pm 3.74}$	$-21.16_{\pm 0.81}$	$-29.05_{\pm 4.66}$	$-24.16_{\pm 4.64}$
CLNode	$80.25_{\pm 2.05}$	$52.31_{\pm 1.36}$	$55.63_{\pm 2.62}$	$-14.70_{\pm 2.43}$	$-49.74_{\pm 1.13}$	$-45.26_{\pm 2.78}$
TSS	$73.99_{\pm 0.55}$	$52.14_{\pm 6.47}$	$49.80_{\pm 6.78}$	$-13.69_{\pm 1.05}$	$-42.47_{\pm 7.24}$	$-40.11_{\pm 10.63}$
TFR	$83.37_{\pm 0.82}$	$60.88_{\pm 7.36}$	$42.42_{\pm 2.64}$	$-8.04_{\pm 3.94}$	$-14.95_{\pm 10.51}$	$-35.46_{\pm 1.84}$
UFO	**$93.36_{\pm 0.01}$**	**$93.29_{\pm 0.08}$**	**$83.10_{\pm 0.89}$**	**$-0.69_{\pm 0.07}$**	**$-0.66_{\pm 0.10}$**	**$-8.27_{\pm 1.16}$**
Photo						
Joint	$96.90_{\pm 0.80}$	$93.24_{\pm 0.72}$	$85.87_{\pm 2.34}$		-	
Bare model	$65.75_{\pm 0.01}$	$49.64_{\pm 0.02}$	$49.63_{\pm 0.05}$	$-38.78_{\pm 0.07}$	$-30.02_{\pm 0.04}$	$-22.34_{\pm 0.06}$
LwF	$80.03_{\pm 19.44}$	$79.21_{\pm 9.37}$	$63.36_{\pm 1.30}$	$-18.51_{\pm 18.04}$	$-12.05_{\pm 19.43}$	$-17.23_{\pm 16.98}$
ERGNN	$95.49_{\pm 0.14}$	$42.62_{\pm 11.31}$	$53.00_{\pm 17.22}$	$-5.37_{\pm 7.26}$	$-0.48_{\pm 1.65}$	$-30.88_{\pm 11.94}$
DSLR	$97.36_{\pm 0.14}$	$87.70_{\pm 1.60}$	$69.62_{\pm 3.95}$	$-1.28_{\pm 0.36}$	$-11.90_{\pm 2.64}$	$-18.56_{\pm 8.21}$
DMSG	$49.58_{\pm 0.02}$	$57.11_{\pm 7.66}$	$50.27_{\pm 0.89}$	$-0.31_{\pm 0.15}$	$-23.91_{\pm 12.93}$	$0.77_{\pm 1.24}$
SCE	$63.85_{\pm 5.86}$	$47.63_{\pm 5.82}$	$57.59_{\pm 11.41}$	$-42.02_{\pm 7.47}$	$-37.85_{\pm 20.60}$	$-10.61_{\pm 15.30}$
CLNode	$89.31_{\pm 9.31}$	$57.28_{\pm 11.90}$	$59.86_{\pm 4.83}$	$-8.53_{\pm 11.61}$	$-50.99_{\pm 14.67}$	$-46.61_{\pm 6.51}$
TSS	$61.44_{\pm 11.38}$	$62.42_{\pm 8.31}$	$55.51_{\pm 10.22}$	$-34.36_{\pm 19.57}$	$-27.04_{\pm 20.64}$	$-35.44_{\pm 4.19}$
TFR	$73.87_{\pm 4.70}$	$51.82_{\pm 4.04}$	$46.42_{\pm 1.78}$	$-28.18_{\pm 6.61}$	$-28.01_{\pm 12.53}$	$-32.90_{\pm 1.15}$
UFO	**$98.07_{\pm 0.04}$**	**$96.04_{\pm 0.25}$**	**$93.40_{\pm 0.86}$**	**$-0.22_{\pm 0.06}$**	**$-0.17_{\pm 0.49}$**	**$1.61_{\pm 0.99}$**

to construct a memory buffer to prevent catastrophic forgetting. For consistency, we store 10 samples per class and use a two-layer GCN as the backbone. Moreover, two standard reference settings are included. Joint is an offline upper-bound reference with access to data from all tasks, while bare model is the backbone GCN without additional strategies, serving as a lower-bound reference. A full description of the baseline implementations is included in Appendix A.2.

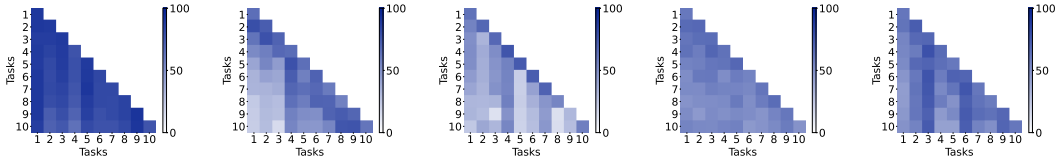
Evaluation Metrics. To evaluate performance under varying noise levels in continual learning, following previous works [20, 53, 54], we use average accuracy and average forgetting as the primary metrics. Accuracy measures the average accuracy over tasks after learning the final task, and Forgetting measures the final retention of previous tasks. A higher accuracy or a higher forgetting

suggests that the model has a better performance. The formal definitions of metrics are given in Appendix A.3.

5.2 Comparison Study

Table 2 shows that UFO performs better on four datasets under varying noise ratios. As the noise ratio increases, all models show lower average accuracy, while forgetting first decreases and then increases. This is because noise may already hurt the model at an early stage, preventing it from learning true knowledge well, so the later performance change becomes less obvious. Existing CGL methods mainly focus on mitigating catastrophic forgetting, so their average forgetting remains relatively stable. On CoraFull, when the noise ratio increases from 0% to 30%, the accuracy of DSLR decreases by 23.17%, while its forgetting increases by 5.49%. However, among graph learning with label noise methods, the forgetting of CLNode and TSS decreases by 50.85% and 37.99%. This may be related to their higher early accuracy, and continuously arriving noisy labels further affect the stability of the model. In contrast, UFO shows smaller changes in both accuracy and forgetting on all four datasets, demonstrating stronger robustness under noisy conditions.

To further analyze the overall behavior of different models, we visualize the performance matrices on CoraFull under the 30% noise setting. As shown in Figure 2, darker colors indicate higher accuracy. For LwF and ERGNN, the dark regions are concentrated along the diagonal and the colors become lighter for earlier tasks. Under noisy labels, DSLR and DMSG show relatively uniform but lighter color distributions. In contrast, UFO shows a more stable and darker matrix, demonstrating that it not only preserves knowledge but also reduces the accumulation of corrupted supervision, which achieves a better balance between catastrophic forgetting and catastrophic remembering. Due to space limitations, more comparisons are provided in Appendix B.2.



(a) UFO (b) LwF (c) ERGNN (d) DSLR (e) DMSG

Figure 2: Performance matrices on CoraFull with 30% noise.

5.3 Ablation Study

To further analyze the effectiveness of each component, we conduct an ablation study on CoraFull and CS with different noise ratios. We consider the bare model (BM), knowledge preservation (KP), relative scores for the new task (NS) and old task (OS), and the replay mechanism (R). The results in Table 3 show that: (1) KP enables the BM to achieve better performance by preserving previous knowledge and reducing feature drift. (2) When NS is further introduced, the model achieves higher accuracy in several noise ratios, especially under 15% noise. (3) R brings clear gains, e.g., improving the forgetting value on CoraFull with 30% noise from -53.92% to -7.93%. (4) After adding OS, UFO achieves the best overall performance across varying noise levels. This suggests that OS helps filter low-quality features generated by the flow model.

5.4 Hyperparameter

We evaluate the sensitivity to the number of flows in Figure 3. As observed, the performance remains stable with 0% noise, indicating a shallow or moderate flow is sufficient for clean distributions. When the noise ratio increases to 30%, accuracy and forgetting reach their best levels around 9–12 flows. With the number of flows further increasing to 13, the results drop clearly. This suggests that a moderate flow depth provides a better balance for robust continual learning, while an overly deep flow increases optimization difficulty.

Table 3: Ablation study of our UFO. The best performance is highlighted in **red**.

Metric	Noise	BM	BM+KP	BM+KP+NS	BM+KP+NS+R	UFO
CoraFull						
Accuracy↑	0%	$33.71_{\pm 4.74}$	$47.14_{\pm 0.40}$	$47.07_{\pm 0.60}$	$53.16_{\pm 2.48}$	$\mathbf{81.78_{\pm 0.85}}$
	15%	$34.02_{\pm 0.58}$	$39.19_{\pm 0.47}$	$40.78_{\pm 0.49}$	$51.21_{\pm 0.12}$	$\mathbf{80.71_{\pm 0.08}}$
	30%	$29.02_{\pm 2.34}$	$33.22_{\pm 1.00}$	$33.27_{\pm 0.46}$	$46.65_{\pm 1.01}$	$\mathbf{77.77_{\pm 1.04}}$
Forgetting↑	0%	$-62.09_{\pm 5.34}$	$-47.37_{\pm 0.46}$	$-47.39_{\pm 0.71}$	$-6.39_{\pm 2.32}$	$\mathbf{-6.35_{\pm 1.10}}$
	15%	$-47.04_{\pm 0.37}$	$-53.46_{\pm 0.62}$	$-51.78_{\pm 0.61}$	$-6.57_{\pm 0.79}$	$\mathbf{-6.33_{\pm 2.25}}$
	30%	$-38.10_{\pm 2.53}$	$-53.98_{\pm 1.17}$	$-53.92_{\pm 0.55}$	$-7.93_{\pm 1.08}$	$\mathbf{-7.10_{\pm 1.27}}$
CS						
Accuracy↑	0%	$61.22_{\pm 0.10}$	$93.53_{\pm 0.54}$	$92.31_{\pm 0.48}$	$96.16_{\pm 0.51}$	$\mathbf{97.29_{\pm 0.29}}$
	15%	$62.83_{\pm 0.07}$	$87.15_{\pm 1.10}$	$89.18_{\pm 0.36}$	$90.32_{\pm 1.12}$	$\mathbf{92.21_{\pm 0.43}}$
	30%	$53.04_{\pm 0.03}$	$69.91_{\pm 3.80}$	$67.11_{\pm 3.60}$	$78.65_{\pm 3.44}$	$\mathbf{81.46_{\pm 0.43}}$
Forgetting↑	0%	$-45.55_{\pm 0.02}$	$-5.59_{\pm 0.62}$	$-7.09_{\pm 0.59}$	$-2.36_{\pm 0.60}$	$\mathbf{-1.30_{\pm 0.27}}$
	15%	$-28.98_{\pm 0.01}$	$-1.99_{\pm 1.25}$	$\mathbf{0.29_{\pm 0.58}}$	$-0.22_{\pm 1.22}$	$0.06_{\pm 1.34}$
	30%	$-25.10_{\pm 0.21}$	$-8.14_{\pm 4.58}$	$-12.28_{\pm 4.20}$	$-0.32_{\pm 4.19}$	$\mathbf{3.14_{\pm 0.20}}$

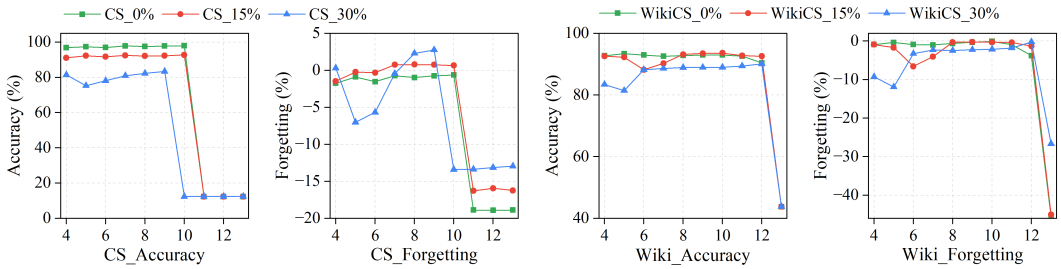
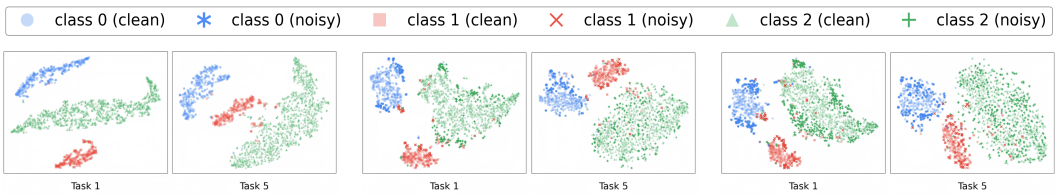


Figure 3: Sensitivity analysis for the number of flows with 0%, 15%, and 30% noise.

5.5 Visualization

To analyze the evolution of clean and noisy nodes, we visualize the embeddings of the classes from the first task (classes 0, 1, and 2) on CS in Figure 4. The three subfigures show the embeddings learned by UFO on sequential CS with 0%, 15%, and 30% noise. The 0% noise setting provides a clean reference, where the embeddings form distinct clusters with minimal overlap. As the noise ratio increases to 15%,30%, we can observe that noisy nodes in UFO are mainly located near the boundary or the outer region of each cluster. Comparing Task 1 and Task 5, the embeddings of old classes show less drift and preserve relatively stable structures. This suggests that UFO can preserve knowledge and maintain robust representations under noisy supervision. The complete visualization results for UFO and Joint are provided in Appendix B.3.



(a) UFO with 0% noise (b) UFO with 15% noise (c) UFO with 30% noise

Figure 4: t-SNE visualization of the evolution of embeddings for classes 0, 1, and 2 on sequential CS.

6 Limitations

This work has two limitations. First, UFO performs replay in the feature space and does not explicitly reconstruct the original graph topology, such as historical edges and neighborhoods. This may limit

the ability of feature replay to fully recover structural knowledge from previous tasks. Second, UFO relies on the configuration of the flow model. An overly deep flow can increase optimization difficulty and computational cost. Thus, the number of flow layers and related hyperparameters need to be selected carefully in practice.

7 Conclusion

In this paper, we study a more realistic robust continual graph learning scenario and present the first investigation of a failure mode in CGL—catastrophic remembering. To jointly address these challenges, we propose UFO, a unified flow-oriented framework. UFO uses normalizing flows to conditionally model feature distributions for replay and estimate instance-level reliability under corrupted supervision. Furthermore, we introduce graph knowledge preservation to constrain structural and semantic consistency. Experimental results demonstrate the superiority of UFO in improving performance and robustness under varying noise levels.

Acknowledgments and Disclosure of Funding

This work was supported by GBA Ascend Application Innovation Institute, Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), under Grant No. GML-ST-2026-24. We would also like to thank reviewers, (S)ACs, and PCs from NeurIPS 2026 for their valuable suggestions and their service to the community.

References

[1] Yongheng Xing, You Wang, Xinqi Du, Juncheng Hu, and Kun Yang. Tapguard: A semantic-aware graph framework for tap rule cascading threat detection. *IEEE Transactions on Information Forensics and Security*, 21:5824–5835, 2026.

[2] Qing Qing, Wang Zhe, Ma Chenglong, Xu Ziqi, Wang Yongjun, Kong Xiangjie, Luo Renqiang, Yu Fengqi, Yang Yuxiao, and Xia Feng. Artificial intelligence in autonomous vehicles: A survey from a fairness perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2026.

[3] Yuanbo Xu, En Wang, Yongjian Yang, and Yi Chang. A unified collaborative representation learning for neural-network based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5126–5139, 2022.

[4] Feng Xia, Ciyuan Peng, Jing Ren, et al. Graph learning. *Foundations and Trends® in Signal Processing*, pages 362–519, 2026.

[5] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*, 2018.

[6] Renqiang Luo, Huafei Huang, Ziqi Xu, Xikun Zhang, Enyan Dai, Bo Yang, and Feng Xia. Fairness in augmented graph learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–18, 2026.

[7] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, pages 3521–3526, 2017.

[8] Yongheng Xing, Liang Hu, Xinqi Du, Zhiqi Shen, Juncheng Hu, and Feng Wang. A privacy-preserving federated graph learning framework for threat detection in iot trigger-action programming. *Expert Syst. Appl.*, 255, December 2024.

[9] Renqiang Luo, Huafei Huang, Shuo Yu, Fengqi Yu, Feng Xia, Sajal K Das, and Chengqi Zhang. Utility-preserving federated graph learning with dual-perspective fairness. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(9):11077–11093, 2026.

[10] Jialu Li, Yu Wang, Pengfei Zhu, et al. What matters in graph class incremental learning? An information preservation perspective. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 26195–26223, 2024.

[11] Hwanjun Song, Minseok Kim, and Jae-Gil Lee. SELFIE: Refurbishing unclean samples for robust deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5907–5915, 2019.

[12] Zhonghao Wang, Danyu Sun, Sheng Zhou, et al. NoisyGL: A comprehensive benchmark for graph neural networks under label noise. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 38142–38170, 2024.

[13] Kunlun Xu, Haozhuo Zhang, Yu Li, et al. Mitigate catastrophic remembering via continual knowledge purification for noisy lifelong person re-identification. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 5790–5799, 2024.

[14] Xikun Zhang, Dongjin Song, and Dacheng Tao. Hierarchical prototype networks for continual graph representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 4622–4636, 2023.

[15] Bo Han, Quanming Yao, Xingrui Yu, et al. Co-teaching: robust training of deep neural networks with extremely noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8536–8546, 2018.

[16] Daiqing Qi, Handong Zhao, Xiaowei Jia, and Sheng Li. Revealing an overlooked challenge in class-incremental graph learning. *Transactions on Machine Learning Research*, 2024.

[17] Xuemei Cao, Hanlin Gu, Xin Yang, et al. ErrorEraser: Unlearning data bias for improved continual learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 119–130, 2025.

[18] Chris Dongjoo Kim, Jinseo Jeong, Sangwoo Moon, et al. Continual learning on noisy data streams via self-purified replay. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 537–547, 2021.

[19] Kailai Li, Jiawei Sun, Jiong Lou, Zhanbo Feng, Hefeng Zhou, Chentao Wu, Guangtao Xue, Wei Zhao, and Jie Li. Leveraging peer-informed label consistency for robust graph neural networks with noisy labels. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 5598–5606. International Joint Conferences on Artificial Intelligence Organization, 8 2025. Main Track.

[20] Xikun Zhang, Dongjin Song, and Dacheng Tao. CGLB: benchmark tasks for continual graph learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 13006–13021, 2022.

[21] Xikun Zhang, Dongjin Song, and Dacheng Tao. Continual learning on graphs: Challenges, solutions, and opportunities. *arXiv preprint arXiv:2402.11565*, 2024.

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017.

[23] Huihui Liu, Yiding Yang, and Xinchao Wang. Overcoming catastrophic forgetting in graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8653–8661, 2021.

[24] Seungyoon Choi, Wonjoong Kim, Sungwon Kim, et al. DSLR: Diversity enhancement and structure learning for rehearsal-based graph continual learning. In *Proceedings of the ACM Web Conference 2024*, pages 733–744, 2024.

[25] Yuanbo Xu, En Wang, Yongjian Yang, and Hui Xiong. GS^2-RS: A generative approach for alleviating cold start and filter bubbles in recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 36(2):668–681, 2024.

[26] En Wang, Mijia Zhang, Bo Yang, Yongjian Yang, and Jie Wu. Large-scale spatiotemporal fracture data completion in sparse crowdsensing. *IEEE Transactions on Mobile Computing*, 23(7):7585–7601, 2024.

[27] Ziyue Qiao, Junren Xiao, Qingqiang Sun, et al. Towards continuous reuse of graph models via holistic memory diversification. In *International Conference on Learning Representations*, 2025.

[28] Mengmeng Sheng, Zeren Sun, Tianfei Zhou, et al. CA2C: A prior-knowledge-free approach for robust label noise learning via asymmetric co-learning and co-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 901–911, 2025.

[29] Hwanjun Song, Minseok Kim, Dongmin Park, et al. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 8135–8153, 2023.

[30] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8792–8802, 2018.

[31] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7017–7025, 2019.

[32] Xingrui Yu, Bo Han, Jiangchao Yao, et al. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pages 7164–7173, 2019.

[33] Yisen Wang, Xingjun Ma, Zaiyi Chen, et al. Symmetric cross entropy for robust learning with noisy labels. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 322–330, 2019.

[34] Scott Reed, Honglak Lee, Dragomir Anguelov, et al. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596*, 2014.

[35] Xiaowen Wei, Xiuwen Gong, Yibing Zhan, et al. CLNode: Curriculum learning for node classification. In *Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, pages 670–678, 2023.

[36] Zhonghao Wang, Yuanchen Bei, Sheng Zhou, et al. Learning from graph: Mitigating label noise on graph through topological feature reconstruction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 3261–3270, 2025.

[37] Yuhao Wu, Jiangchao Yao, Xiaobo Xia, et al. Mitigating label noise on graphs via topological sample selection. In *Proceedings of the 41st International Conference on Machine Learning*, pages 53944–53972, 2024.

[38] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

[39] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.

[40] Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. Semi-supervised learning with normalizing flows. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4615–4630, 13–18 Jul 2020.

[41] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, et al. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, (57):1–64, 2021.

[42] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

[43] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[45] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035, 2017.

[46] Abudukelimu Wuerkaixi, Sen Cui, Jingfeng Zhang, et al. Accurate forgetting for heterogeneous federated continual learning. In *The 12th International Conference on Learning Representations*, 2024.

[47] Enyan Dai, Charu Aggarwal, and Suhang Wang. NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 227–236, 2021.

[48] Kuan Zhang, Chengliang Chai, Jingzhe Xu, et al. Handling label noise via instance-level difficulty modeling and dynamic optimization. In *Proceedings of the 39th International Conference on Neural Information Processing Systems*, pages 46667–46696, 2025.

[49] Yiding Yang, Jiayan Qiu, Mingli Song, et al. Distilling knowledge from graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7074–7083, 2020.

[50] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[51] Péter Mernyei and Cătălina Cangea. Wiki-CS: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

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

[53] Ting Wu, Jingyi Liu, Rui Zheng, et al. Enhancing contrastive learning with noise-guided attack: Towards continual relation extraction in the wild. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2227–2239, 2024.

[54] Yinlin Zhu, Miao Hu, and Di Wu. Federated continual graph learning. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 4203–4213, 2025.

A Implementation Details

A.1 Datasets

In our experiments, we evaluate our method on four widely used graph datasets.

- **CoraFull** [5] is an academic citation network with 70 classes, where nodes denote academic papers and edges denote citation links between papers.
- **CS** [50] is an author collaboration network extracted from the Microsoft Academic Graph. In this dataset, each node represents an author, and an edge connects two authors who have co-authored a paper.
- **WikiCS** [51] is built from Wikipedia on computer science topics. It treats articles as nodes and hyperlinks as edges, and provides multiple predefined splits for reliable evaluation.
- **Photo** [50] is an Amazon co-purchase network from the photography category. Nodes represent products, and an edge is created when two items are frequently purchased together, reflecting implicit product relationships.

The statistics of these datasets are summarized in Table 4.

Table 4: Statistics of the Datasets.

Dataset	#Nodes	#Edges	#Features	#Classes	#Tasks
CoraFull	19,793	126,842	8,710	70	10
CS	18,333	163,788	6,805	15	5
WikiCS	11,701	431,726	300	10	5
Photo	7,650	238,162	745	8	4

A.2 Baselines

We compare our method with several state-of-the-art approaches from both continual graph learning and graph learning with label noise. The comparison methods are briefly introduced as follows.

- **Joint** trains the backbone GCN on all tasks simultaneously. Since it has access to the full task sequence, it is reported only as an upper-bound reference. This setting removes the influence of catastrophic forgetting and highlights the negative effect of noisy supervision.
- **Bare model** is the backbone GCN without any continual learning or robust strategy. It serves as a lower-bound reference, where the model is directly affected by catastrophic forgetting and catastrophic remembering.

Continual graph learning (adding an SCE loss term weighted by 0.5):

- **LwF** [22] is a regularization-based method that preserves knowledge from previous tasks by distilling soft targets from a historical model, which can be seen as a hybrid of knowledge distillation and fine-tuning.
- **ERGNN** [52] proposes experience node selection strategies from previous tasks, including mean of feature, coverage maximization, and influence maximization, and replays these representative nodes to mitigate catastrophic forgetting when learning new tasks.
- **DSLR** [24] improves rehearsal-based approaches by considering coverage-based diversity and graph structure learning, which helps preserve high-quality information and mitigate catastrophic forgetting.
- **DMSG** [27] incorporates a buffer selection strategy that considers both intra-class and inter-class diversities, and includes a generative memory replay module, thereby preserving comprehensive knowledge and enhancing memory retention.

Graph learning with label noise (replaying 10 nodes per class):

- **SCE** [33] is a robust loss function that combines cross-entropy with reverse cross-entropy, simultaneously addressing both overfitting and under learning under label noise.

- **CLNode** [35] uses curriculum learning for graph data, where node difficulty is estimated from multiple perspectives and the model is trained from easier to harder samples.
- **TSS** [37] leverages graph topology to initially select high-confidence nodes, allowing the model to learn from more reliable samples under label noise. It uses class-conditional betweenness centrality to gradually incorporate clean, informative nodes near class boundaries.
- **TFR** [36] introduces a dual-network framework with a backbone GNN and a decoder GNN to reconstruct the feature distribution. It also guides the selection of high-confidence pseudo-labels, providing additional supervision beyond the observed noisy labels.

A.3 Evaluation Metrics

We adopt two commonly used metrics in continual learning, namely accuracy and forgetting [20], to evaluate both the overall performance and robustness of the model across tasks. Formally, they are defined as:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} A_{n,i}, \quad \text{Forgetting} = \frac{1}{n-1}\sum_{i=1}^{n-1}\left(A_{n,i} - A_{i,i}\right), \tag{15}$$

where n denotes the total number of tasks, and $A_{i,j}$ represents the test accuracy on task j after training up to task i. In particular, $A_{i,i}$ is the accuracy immediately after learning task i, while $A_{n,i}$ denotes its final accuracy after all tasks are learned.

A.4 Details Settings

We adopt the GCN as the backbone, with a hidden dimension of 256. Following [17, 20], the model is trained using the Adam optimizer with a learning rate of 0.005 for 200 epochs on each task. The flow model consists of 4 affine coupling layers and 4 permutation layers. The temperature parameter is set to $\tau = 2.71$, with $\alpha_E = 0.09$ and $\alpha_L = 0.06$. All results are reported as the mean and standard deviation over three independent runs, with implementation based on PyTorch 2.3.0 and CUDA 12.1 on a 48GB vGPU.

B Additional Experimental Results

B.1 Pair Flipping Noise

In many scenarios, noisy labels are not randomly assigned to any class, but are more likely to be confused with very similar classes. For example, in the CS dataset [50], authors in paired research fields, such as machine learning and data mining or robotics and computer vision, may share similar research topics. To simulate this, we further conduct experiments under pair flipping noise with ratios of 0%, 15%, and 30%. The results are reported in Table 5, and it can be observed that UFO remains effective in preserving performance under structured label corruption.

Table 5: Average performance comparison on CS under pair flipping noise. The best performance is in **red**, and the second best is in blue.

Models	Accuracy↑			Forgetting↑		
	0%	15%	30%	0%	15%	30%
CS						
Joint	$97.00_{\pm 0.10}$	$82.15_{\pm 3.06}$	$73.01_{\pm 4.73}$		-	
Bare model	$61.22_{\pm 0.10}$	$43.14_{\pm 5.24}$	$41.59_{\pm 10.05}$	$-45.55_{\pm 0.02}$	$-43.15_{\pm 6.05}$	$-31.36_{\pm 11.32}$
LwF	$92.51_{\pm 4.10}$	$57.48_{\pm 5.07}$	$62.28_{\pm 6.75}$	$-5.49_{\pm 4.78}$	$-30.89_{\pm 6.60}$	$-4.27_{\pm 7.36}$
ERGNN	$94.40_{\pm 0.14}$	$61.32_{\pm 3.09}$	$60.26_{\pm 3.97}$	$-3.26_{\pm 0.29}$	$-29.69_{\pm 8.80}$	$-15.50_{\pm 2.88}$
DSLR	$96.47_{\pm 0.80}$	$81.85_{\pm 1.56}$	$65.78_{\pm 2.70}$	$-2.01_{\pm 0.88}$	$-6.70_{\pm 2.16}$	$-5.04_{\pm 4.40}$
DMSG	$93.87_{\pm 1.14}$	$68.28_{\pm 1.72}$	$74.68_{\pm 3.88}$	$-4.24_{\pm 1.43}$	$-15.91_{\pm 4.15}$	$\mathbf{4.32_{\pm 8.61}}$
SCE	$78.90_{\pm 7.20}$	$48.26_{\pm 14.97}$	$54.99_{\pm 8.05}$	$-22.48_{\pm 8.69}$	$-52.39_{\pm 9.29}$	$-27.48_{\pm 17.28}$
CLNode	$93.84_{\pm 0.32}$	$49.72_{\pm 1.92}$	$48.17_{\pm 1.45}$	$-3.84_{\pm 0.47}$	$-57.87_{\pm 2.50}$	$-58.73_{\pm 1.96}$
TSS	$79.62_{\pm 5.16}$	$48.28_{\pm 4.03}$	$49.41_{\pm 12.24}$	$-14.29_{\pm 8.06}$	$-47.73_{\pm 2.55}$	$-31.95_{\pm 11.31}$
TFR	$94.11_{\pm 1.98}$	$39.92_{\pm 10.13}$	$42.09_{\pm 3.92}$	$-3.43_{\pm 2.58}$	$-50.30_{\pm 11.79}$	$-38.35_{\pm 8.88}$
UFO	$\mathbf{97.29_{\pm 0.29}}$	$\mathbf{91.33_{\pm 0.32}}$	$\mathbf{77.70_{\pm 1.83}}$	$\mathbf{-1.30_{\pm 0.27}}$	$\mathbf{0.81_{\pm 0.31}}$	$-0.84_{\pm 1.79}$

B.2 Performance Matrices

Figures 5–7 present additional performance matrices on CoraFull under different noise ratios. As can be seen, most baselines show clear degradation on earlier tasks as the task sequence progresses, and this degradation becomes more evident when the noise ratio increases. Although some methods, such as SCE, CLNode, TSS, and TFR, produce darker diagonal regions, the lighter lower-left regions indicate limited preservation of previous task knowledge under noisy supervision. This suggests that improving robustness to noisy labels alone is not sufficient for robust continual graph learning, which further supports the need for a unified framework such as UFO to handle knowledge preservation and noisy supervision.

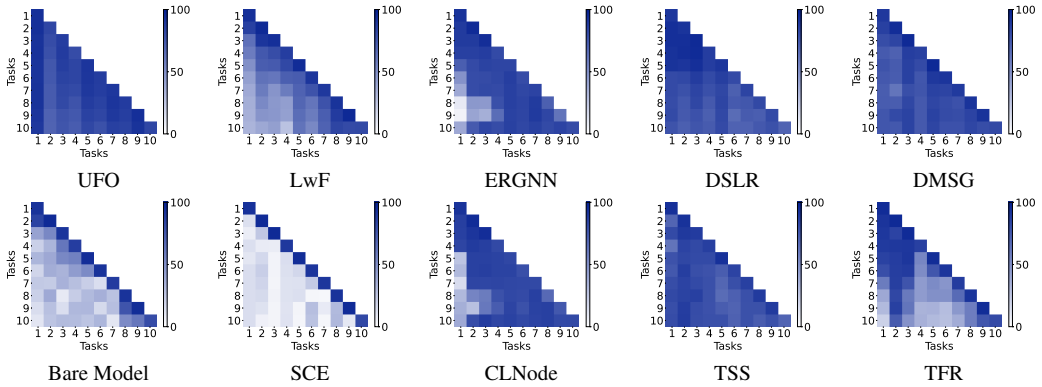


Figure 5: Performance matrices of various methods on CoraFull with 0% noise.

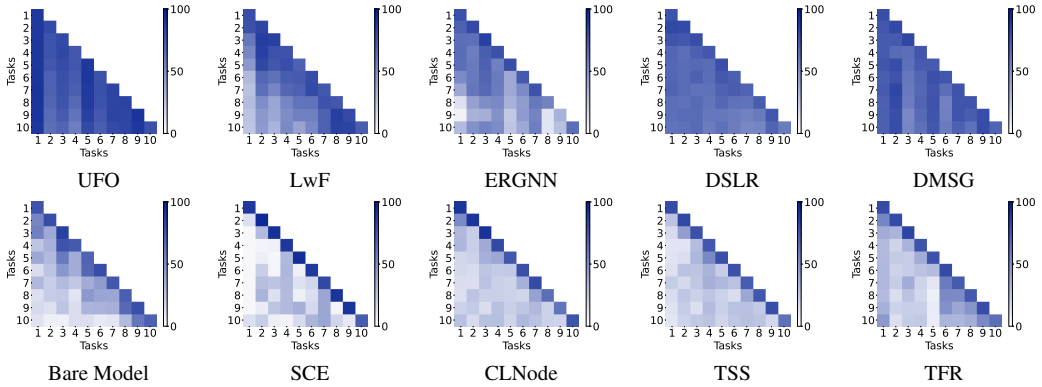


Figure 6: Performance matrices of various methods on CoraFull with 15% noise.

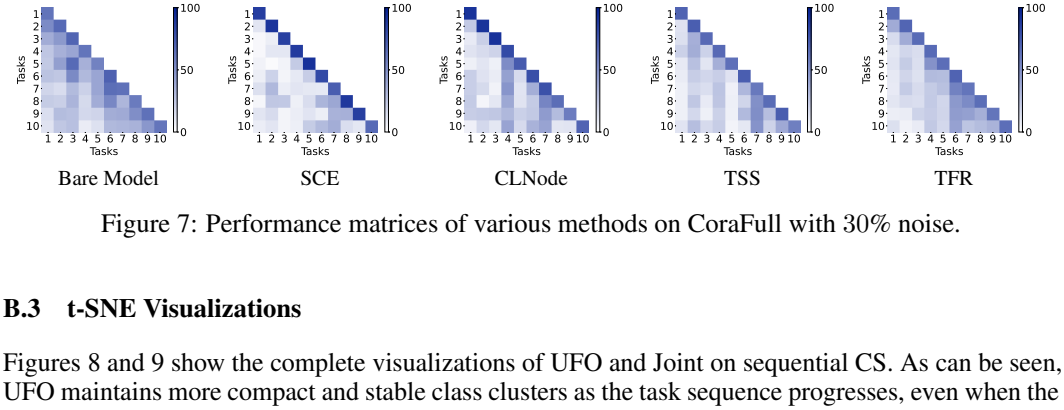
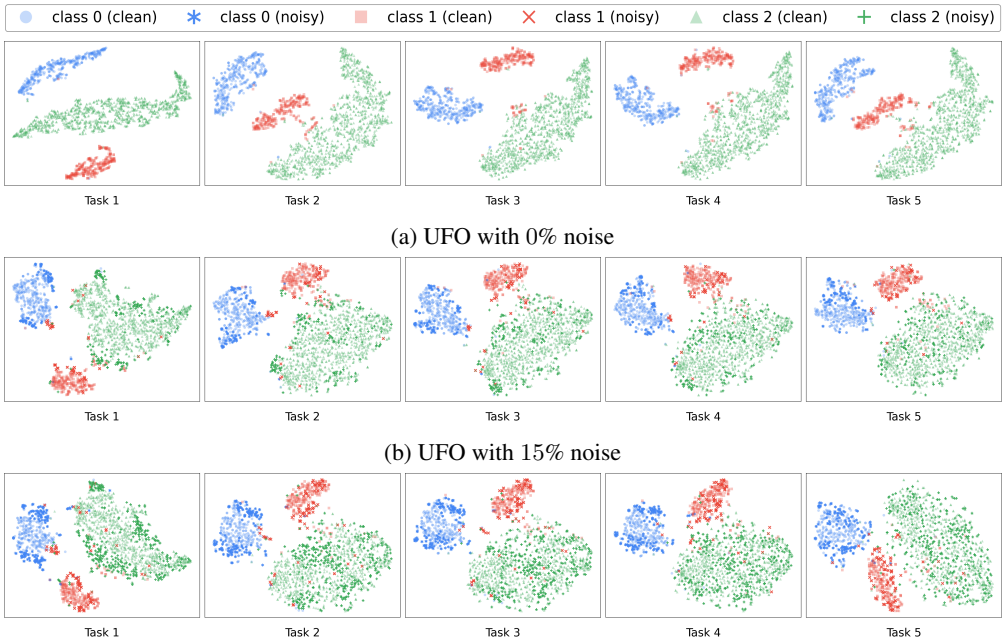


Figure 7: Performance matrices of various methods on CoraFull with 30% noise.

B.3 t-SNE Visualizations

Figures 8 and 9 show the complete visualizations of UFO and Joint on sequential CS. As can be seen, UFO maintains more compact and stable class clusters as the task sequence progresses, even when the

noise ratio increases. Noisy nodes are mainly distributed near the cluster boundaries or outer regions, indicating that UFO can reduce their disturbance to the main class structure. In contrast, Joint shows more scattered embeddings under noisy supervision, especially at higher noise ratios, where the class structures become less clear. These visualizations further demonstrate the effectiveness of UFO.

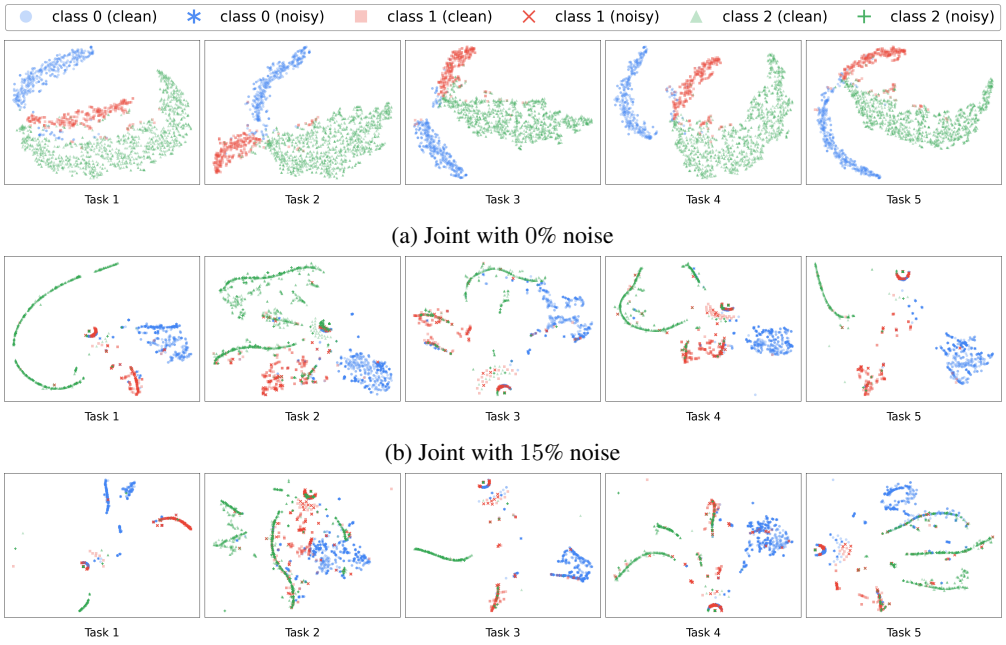


(a) UFO with 0% noise

(b) UFO with 15% noise

(c) UFO with 30% noise

Figure 8: t-SNE visualization of the evolution of embeddings for classes 0, 1, and 2 on sequential CS.



(a) Joint with 0% noise

(b) Joint with 15% noise

(c) Joint with 30% noise

Figure 9: t-SNE visualization of the evolution of embeddings for classes 0, 1, and 2 on sequential CS.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both the abstract and introduction clearly state the problem setting, main contributions, and scope of the paper. See Section 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of feature replay and the sensitivity to the flow configuration in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not include assumptions and proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 specifies the datasets, task splits, noise settings, baselines, and evaluation metrics. Appendix A provides the implementation details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets (CoraFull, CS, WikiCS, Photo) used are publicly available. We provide the code and data as supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 5.1 provides the data splits, noise, baselines, and evaluation metrics. Appendix A provides the implementation details. Code is provided in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For all main experiments, we report mean values with standard deviations across different datasets and noise ratios.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The computational resources and running environment are provided in appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that our work follows the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This work aims to improve the reliability and robustness of CGL. We do not foresee significant direct negative societal impacts.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper does not release pre-trained language models, image generators, or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All code, datasets, and models are cited and used under their respective licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code is provided in the supplementary material with documentation and reproduction instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not used for developing the core ideas or methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.