# Overcoming Catastrophic Forgetting in Graph Neural Networks with Experience Replay

**Fan Zhou, Chengtai Cao***
University of Electronic Science and Technology of China
fan.zhou@uestc.edu.cn, cct447131988@gmail.com

## Abstract

Graph Neural Networks (GNNs) have recently received significant research attention due to their superior performance on a variety of graph-related learning tasks. Most of the current works focus on either static or dynamic graph settings, addressing a single particular task, e.g., node/graph classification, link prediction. In this work, we investigate the question: can GNNs be applied to *continuously* learning a sequence of tasks? Towards that, we explore the Continual Graph Learning (CGL) paradigm and present the **E**xperience **R**eplay based framework **ER-GNN** for CGL to alleviate the catastrophic forgetting problem in existing GNNs. ER-GNN stores knowledge from previous tasks as experiences and replays them when learning new tasks to mitigate the catastrophic forgetting issue. We propose three experience node selection strategies: *mean of feature*, *coverage maximization*, and *influence maximization*, to guide the process of selecting experience nodes. Extensive experiments on three benchmark datasets demonstrate the effectiveness of our ER-GNN and shed light on the incremental graph (non-Euclidean) structure learning.

## Introduction

Applying deep learning methods for graph data analytics tasks has recently generated a significant research interest (Wu et al. 2019b). Plenty of models have been developed to tackle various graph-related learning tasks, including node classification, link prediction, broader graph classifications, etc. Earlier efforts (Perozzi, Al-Rfou, and Skiena 2014; Tang et al. 2015; Grover and Leskovec 2016) mainly focused on encoding nodes in networks/graphs[1] into a low-dimensional vector space, while preserving both the topological structure and the node attribute information in an unsupervised manner. However, researchers have recently shifted from developing sophisticated deep learning models on Euclidean-like domains (e.g., image, text) to non-Euclidean graph structure data. This, in turn, resulted in many notable Graph Neural Networks (GNNs) – e.g., GCN (Kipf and Welling 2017), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAT (Veličković et al. 2018), SGC (Wu et al. 2019a), and GIN (Xu et al. 2019).

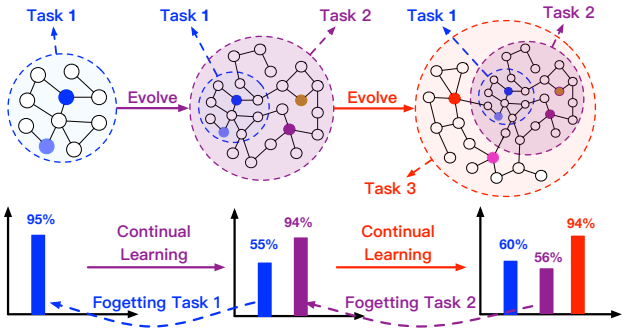



Figure 1: A CGL example. Nodes are papers and links indicate there is a citation between two papers. The task is to classify each node into several predefined classes (e.g., topics). From left to right is the evolution of the citation network. The histograms in different colors denote the node classification performance for each corresponding task.

Despite significant breakthroughs achieved in GNNs, existing models – in both static and dynamic graph settings – primarily focus on a single task. Learning multiple tasks in sequence remains a fundamental challenge for GNNs. A natural question is how do these popular GNNs perform on learning a series of graph-related tasks, which is termed as *Continual Graph Learning* (CGL) in this paper. Take Cora citation data as an illustrative scenario, as shown in Figure 1. In the beginning, there are several nodes in the citation network representing some papers belong to a set of classes. We can train a node classification model on the current graph. However, the real-word citation network is naturally evolving overtime. Consequently, a group of new nodes from new classes (some are labeled and the others are unlabeled) will be added into the graph. We expect that the same model can classify the new nodes. Over time, multiple groups of nodes have been added to the network at different time-points. We term classifying each group of nodes as a task and expect to learn a common node classifier across all tasks, rather than one for the entire graph. Towards that, we train the classifier using each set of nodes in a sequential way. However, this kind of training process can easily lead to the phe-

*Corresponding author

[1]Whenever there is no ambiguity, the terms networks and graphs will be used interchangeably throughout this paper.

nomenon known as catastrophic forgetting (cf. the bottom of Figure 1), where the classifier is updated and overwritten after learning a new task – which is likely to result in a significant drop on the classification performance of previous tasks. Note that the classes/labels in one task are different from those in other tasks. Therefore, this learning process is often perceived as task-incremental learning (De Lange et al. 2019), which has practical value for many real-world graph-related applications, such as web-scale recommender systems (Ying et al. 2018), traffic condition predictions (Chen et al. 2019), and protein design (Ingraham et al. 2019).

Continual learning, also referred to as lifelong learning, sequential learning, or incremental learning, has recently drawn significant research attention. Its objective is to gradually extend the acquired knowledge for future learning, which is very similar to human intelligence (Chen and Liu 2016). Continual learning focuses on learning multiple tasks sequentially, targeting at two general goals: (i) learning a new task does not lead to catastrophic forgetting of former tasks (Goodfellow et al. 2013) and (ii) the model can leverage knowledge from prior tasks to facilitate the learning of new tasks. Catastrophic forgetting is a direct outcome of a more general problem in neural networks, the so-called "stability-plasticity" dilemma (Grossberg 2012). While stability indicates the preservation of previously acquired knowledge, plasticity refers to the ability to integrate new knowledge. This stability-plasticity trade-off is an essential aspect of both artificial and biological neural intelligent systems. Existing studies in continual learning mainly focus on image classification and reinforcement learning tasks, which have yielded several successful methods – e.g., iCaRL (Rebuffi et al. 2017), GEM (Lopez-Paz and Ranzato 2017), EWC (Kirkpatrick et al. 2017), SI (Zenke, Poole, and Ganguli 2017), LwF (Li and Hoiem 2017), and PackNet (Mallya and Lazebnik 2018). However, despite the extensive studies and promising results, there are surprisingly few works on CGL. The three major reasons are: (i) graph (non-Euclidean data) is not independent and identically distributed data; (ii) graphs can be irregular, noisy and exhibit more complex relations among nodes; and (iii) apart from the node feature information, the topological structure in graph plays a crucial role in addressing graph-related tasks.

To bridge this gap, in this work, we target at solving the continual learning problem for graph-structured data through formulating a continual node classification problem. We also conduct an empirical investigation of catastrophic forgetting in GNNs. To our knowledge, we are among the first to analyze graph data in such a sequential learning setting. We present a novel and general **E**xperience **R**eplay **GNN** framework (**ER-GNN**) which stores a set of nodes as experiences in a buffer and replays them in subsequent tasks, providing the capability of learning multiple consecutive tasks and alleviating catastrophic forgetting. For the experience selection, besides two intuitive strategies, we propose a novel scheme built upon influence function (Hampel et al. 2011; Koh and Liang 2017), which performed quite favorably in our evaluation. In summary, we make the following contributions:

- We present the continual graph learning (CGL) paradigm and formulate a new continual learning problem for node classification. The main difference from previous GNN works is that we aim to learn multiple consecutive tasks rather than a single task.
- We conduct an empirical investigation of the continual node classification task, demonstrating that existing GNNs are in the dilemma of catastrophic forgetting when learning a stream of tasks in succession.
- To address the catastrophic forgetting issue, we develop a generic experience replay based framework that can be easily combined with any popular GNNs model. Apart from two intuitive experience selection schemes, we propose a novel strategy based on influence function.
- We conduct extensive experimental evaluations using three benchmarks to demonstrate our framework's superiority over several state-of-the-art GNNs.

## Related Work

### Graph Neural Networks

Graph neural networks have recently become powerful models for learning representations of graphs. Many excellent GNN models have been proposed to exploit the structural information underlying graphs. They also potentially benefit many real-world applications, ranging from node classification and link prediction to traffic prediction and better recommendation (Kipf and Welling 2017; Gao, Wang, and Ji 2018; Wu et al. 2019a; Zhang et al. 2019). Most of the current GNNs can be categorized into two groups: spatial and spectral methods. Spatial methods aggregate the node representations directly from its neighborhood (Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018; Wang et al. 2019), while the basic idea behind spectral approaches is to learn graph representation in the spectral domain where the learned filters are based on Fourier transformation (Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2017). Although these GNNs have achieved great success in many graph-related applications, they only learn a single task. That is, they cannot be generalized to scenarios that require continuous learning while maintaining the model performance on previous tasks. In this work, we study a novel but fundamental problem in graph learning, i.e., how to train a GNN on a sequence of tasks, each of which is a typical graph-related problem such as node classification. Most importantly, the learned GNNs can successfully overcome the forgetting issue and allow us to retrospect earlier model behavior.

### Continual Learning

Several approaches have been proposed to tackle catastrophic forgetting over the last few years. We can roughly distinguish three lines of work: (i) experience replay based methods; (ii) regularization-based methods; (iii) parameter isolation based methods. The first line of works stores samples in their raw format or compressed in a generative model. The stored samples from previous tasks are replayed when learning new tasks without significant forgetting. These samples/pseudo-samples can be used either for

rehearsal – approximating the joint training of previous and current tasks – or to constrain the optimization (Lopez-Paz and Ranzato 2017; Rebuffi et al. 2017). The second line of works proposes an additional regularization term in the loss function to consolidate previous knowledge when new data are available (Kirkpatrick et al. 2017; Li and Hoiem 2017; Zenke, Poole, and Ganguli 2017). The last line of works attempts to prevent any possible forgetting of the previous tasks via models where different parameter subsets are dedicated to different tasks. When there is no constraint on the architecture's scale, it can be done by freezing the set of parameters learned after each previous task and growing new branches for new tasks. Alternatively, under a fixed architecture, methods proceed by identifying the parts used for the earlier tasks and masking them out during the training of the new task (Mallya and Lazebnik 2018; Mallya, Davis, and Lazebnik 2018; Xu and Zhu 2018). These methods have achieved great success in image classification and reinforcement learning tasks. However, they have not been investigated on graph-structured data, which motivates our study in this paper. Our method belongs to the family of experience replay based methods. Furthermore, we propose a new experience selection strategy based on influence function (Hampel et al. 2011; Koh and Liang 2017), in addition to two intuitive experience selection schemes.

## Methodology

This section describes the details of our proposed general framework ER-GNN for continual node classification. We begin with the formal definition of our problem and the biological theory of experience replay based methods, followed by the details of our ER-GNN where three experience selection strategies are presented.

### Problem Definition

The settings of Continual Node Classification (i.e., task incremental learning) problem assume the existence of a collection of tasks: $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_M\}$ which are encountered sequentially and each $\mathcal{T}_i \in \mathcal{T}(|\mathcal{T}| = M)$ is a node classification task. Formally, the node classification task is defined as:

**Definition 1 (Node Classification)** *For each task $\mathcal{T}_i$, we have training node set $\mathcal{D}_i^{tr}$ and testing node set $\mathcal{D}_i^{te}$. Node classification aims to learn a task-specific classifier on $\mathcal{D}_i^{tr}$ that is excepted to classify each node in $\mathcal{D}_i^{te}$ into correct class $(y_i^l \in \mathcal{Y}_i)$, where $\mathcal{Y}_i = \{y_i^1, y_i^2, \ldots, y_i^l, \ldots, y_i^L\}$ is the label set and $L$ is the number of classes in task $\mathcal{T}_i$.*

In the continual graph learning setting, instead of focusing on a single task $\mathcal{T}_i$, we need to learn a series of node classification task set $\mathcal{T}$. That is, our goal is to learn a model $f_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$ that can learn these tasks successively. In particular, we expect the classifier $f_{\boldsymbol{\theta}}$ to not only perform well on the current task but also overcome catastrophic forgetting with respect to the previous tasks.

### Biological Theory

Complementary Learning Systems (CLS) is a well-supported model of biological learning in human beings. It suggests that neocortical neurons learn with an algorithm that is prone to catastrophic forgetting. The neocortical learning algorithm is complemented by a virtual experience system that replays memories stored in the hippocampus to continually reinforce tasks that have not been recently performed (McClelland, McNaughton, and O'Reilly 1995; Kumaran, Hassabis, and McClelland 2016). The CLS theory defines the complementary contribution of the hippocampus and the neocortex in learning and memory, suggesting that there are specialized mechanisms in the human cognitive system for protecting consolidated knowledge. The hippocampal system exhibits short-term adaptation and allows for the rapid learning of new information, which will, in turn, be transferred and integrated into the neocortical system for its long-term storage.

**Algorithm 1** Framework of our ER-GNN.

**Input**: Continual tasks $\mathcal{T}$: $\{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_i, \ldots, \mathcal{T}_M\}$; Experience buffer: $\mathbb{B}$; Number of examples in each class added to $\mathbb{B}$: $e$.
**Output**: Model $f_{\boldsymbol{\theta}}$ which can mitigate catastrophic forgetting of preceding tasks.

```
Initialize θ at random;
while continual task 𝒯 remains do
    Obtain training set 𝒟_i^tr from current task 𝒯_i
    Extract experience nodes B from experience buffer 𝔹
    Compute loss function: ℒ'_{𝒯_i}(f_θ, 𝒟_i^tr, B)
    Compute optimal parameters:
    θ = arg min_{θ∈Θ}(ℒ'_{𝒯_i}(f_θ, 𝒟_i^tr, B))
    Select experience nodes ℰ = Select(𝒟_i^tr, e)
    Add ℰ to experience buffer: 𝔹 = 𝔹 ∪ ℰ
    𝒯 = 𝒯 \ {𝒯_i}
end while
Return model f_θ
```

### Experience Node Replay

Inspired by the CLS theory, we propose a novel and general framework dubbed ER-GNN that selects and preserves experience nodes from the current task and replays them in future tasks. The framework of our ER-GNN is outlined in Algorithm 1.

When learning a task $\mathcal{T}_i$, we acquire its training set $\mathcal{D}_i^{\text{tr}}$ and testing set $\mathcal{D}_i^{\text{te}}$. Subsequently, we select examples $B$ from the experience buffer $\mathbb{B}$. Then we feed the training set $\mathcal{D}_i^{\text{tr}}$ and the experience nodes $B$ together to our classifier $f_{\boldsymbol{\theta}}$. A natural loss function choice for node classification task is the cross-entropy loss function:

$$\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}) = -\Big(\sum_{(\boldsymbol{x_i}, y_i)\in\mathcal{D}} (y_i \log f_{\boldsymbol{\theta}}(\boldsymbol{x_i}) + (1-y_i)\log(1-f_{\boldsymbol{\theta}}(\boldsymbol{x_i})))\Big). \quad (1)$$

Note that the number of nodes in training set $\mathcal{D}_i^{\text{tr}}$ is usually significantly larger than the size of the experience buffer. Here we need a weight factor $\beta$ to balance the influence from $\mathcal{D}_i^{\text{tr}}$ and $B$, averting the model from favoring a particular set

of nodes. According to our empirical observations, we design a dynamic weight factor mechanism:

$$\beta = |B|/(|\mathcal{D}_i^{\text{tr}}| + |B|), \tag{2}$$

where $|\mathcal{D}_i^{\text{tr}}|$ and $|B|$ are the number of nodes in the training set of $\mathcal{T}_i$ and the size of current experience buffer, respectively. The basic idea of this choice is to dynamically balance the following two losses:

$$\begin{aligned}\mathcal{L}'_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}_i^{\text{tr}}, B) =& \beta \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}_i^{\text{tr}}) \\ &+ (1-\beta)\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, B).\end{aligned} \tag{3}$$

Subsequently, we perform parameter updates to obtain the optimal parameters by minimizing the empirical risk:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}\in\Theta}(\mathcal{L}'_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}_i^{\text{tr}}, B)), \tag{4}$$

which can be trained with any optimization methods such as Adma optimizer (Kingma and Ba 2014). After updating the parameters we need to select certain nodes in $\mathcal{D}_i^{\text{tr}}$ as experience nodes $\mathcal{E}$ and add them into the experience buffer $\mathbb{B}$. $\mathcal{E} = \text{Select}(\mathcal{D}_i^{\text{tr}}, e)$ means choosing $e$ nodes in each class as experiences of this task which will be cached into experience buffer $\mathbb{B}$.

The experience selection strategy is crucial to the performance of CGL. We now turn our attention to the problem of identifying which nodes should be stored in the experience buffer $\mathbb{B}$. In the sequel, we present three schemes based on *mean of feature*, *coverage maximization*, and *influence maximization*.

**Mean of Feature (MF)**: Intuitively, the most representative nodes in each class are the ones closest to the average feature vector. Similar to the prior work (Rebuffi et al. 2017) on continual image classification, for each task we compute a prototype for each class and choose $e$ nodes that are the first $e$ nodes closest to this prototype to form the experiences. In some attributed networks, each node has its own attribute vector $\boldsymbol{x_i}$ and embedding vector $\boldsymbol{h_i}$. This means we can obtain our prototypes based on the average attribute vector or the average embedding vector. Therefore, in the MF scheme, we compute the mean of attribute/embedding vector to produce a prototype and choose $e$ nodes whose attribute/embedding vectors are closest to the prototype:

$$\mathbf{c}_l = \frac{1}{|S_l|}\sum_{(\boldsymbol{x_i}, y_i)\in S_l} \boldsymbol{x_i}, \mathbf{c}_l = \frac{1}{|S_l|}\sum_{(\boldsymbol{x_i}, y_i)\in S_l} \boldsymbol{h_i}, \tag{5}$$

where $S_l$ is the set of training nodes in class $l$ and $\mathbf{c}_l$ is the prototype of nodes in class $l$. It is worth noting that although we can calculate prototypes from embedding vectors, we save the original nodes (i.e., $\boldsymbol{x_i}$) as our experiences since we will feed these nodes to our model again when learning new tasks.

**Coverage Maximization (CM)**: When the number of experience nodes $e$ in each class is small, it might be helpful to maximize the coverage of the attribute/embedding space. Drawing inspiration from the prior work (de Bruin et al. 2016) on continual reinforcement learning, we hypothesize that approximating a uniform distribution over all nodes from the training set $\mathcal{D}_i^{\text{tr}}$ in each task $\mathcal{T}_i$ can facilitate choosing experience nodes. To maximize the coverage of the attribute/embedding space, we rank the nodes in each class according to the number of nodes from other classes in the same task within a fixed distance $d$:

$$\mathcal{N}(v_i) = \{v_j | \text{dist}(v_i - v_j) < d, \mathcal{Y}(v_i) \neq \mathcal{Y}(v_j)\}, \tag{6}$$

where $\mathcal{Y}(v_i)$ is the label of node $v_i$, $\mathcal{N}(v_i)$ is the set of nodes coming from different classes within distance $d$ to $v_i$. We can choose $e$ nodes with the lowest $|\mathcal{N}(v_i)|$ in each class as our experiences. Similarly to MF, we can maximize the coverage of either the attribute space or the embedding space, but only store the original nodes as experience.

**Influence Maximization (IM)**: When training on each task $\mathcal{T}_i$, we can remove one training node $v_\star$ from the training set $\mathcal{D}_i^{\text{tr}}$ and obtain a new training set $\mathcal{D}_{i\star}^{\text{tr}}$. Then we can calculate the optimal parameters $\boldsymbol{\theta}_\star$ as:

$$\boldsymbol{\theta}_\star = \arg\min_{\boldsymbol{\theta}\in\Theta}(\mathcal{L}'_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}_{i\star}^{\text{tr}}, B)), \tag{7}$$

resulting in a change in model optimal parameters: $\boldsymbol{\theta}_\star - \boldsymbol{\theta}$. However, obtaining the influence of every removed training node $v_\star$ is prohibitively expensive since it requires retraining the model for each removed node. Fortunately, influence function (Hampel et al. 2011) provides theoretical foundations for estimating the change of parameters without model retraining and has been successfully used in previous works (Koh and Liang 2017) for explaining the behaviors of neural networks. The basic idea is to compute the change of optimal parameters if $v_\star$ was upweighted by some small $\epsilon$, which gives the new parameters:

$$\boldsymbol{\theta}_{\boldsymbol{\epsilon},\star} \overset{\text{def}}{=} \arg\min_{\boldsymbol{\theta}\in\Theta}(\mathcal{L}'_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, \mathcal{D}_i^{\text{tr}}, B) + \epsilon\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_\star)), \tag{8}$$

where the influence of upweighting $v_\star$ on the parameters $\boldsymbol{\theta}$ is given by :

$$\mathcal{I}_{\text{up},\boldsymbol{\theta}}(v_\star) \overset{\text{def}}{=} \left.\frac{\partial \boldsymbol{\theta}_{\boldsymbol{\epsilon},\star}}{\partial \epsilon}\right|_{\epsilon=0} = -\mathbf{H}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_\star), \tag{9}$$

where $\mathbf{H}_{\boldsymbol{\theta}}$ is the Hessian matrix that can be computed as:

$$\mathbf{H}_{\boldsymbol{\theta}} \overset{\text{def}}{=} \frac{1}{(|\mathcal{D}_i^{\text{tr}}| + |B|)}\sum_{j=1}^{(|\mathcal{D}_i^{\text{tr}}|+|B|)} \nabla^2_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_j), \tag{10}$$

and Eq.(8) suggests that removing node $v_\star$ is the same as upweighting it by $\epsilon = -(1/(|\mathcal{D}_i^{\text{tr}}| + |B|))$. Thus, we can linearly approximate the parameter change of removing $v_\star$ as $\boldsymbol{\theta}_\star - \boldsymbol{\theta} \approx -(1/(|\mathcal{D}_i^{\text{tr}}| + |B|))\mathcal{I}_{\text{up},\boldsymbol{\theta}}(v_\star)$, without retraining the model. However, the Frobenius norm of $\boldsymbol{\theta}_\star - \boldsymbol{\theta}$ is usually too small to find the exact $\boldsymbol{\theta}_\star$. Besides, calculating the inverse of matrix $\mathbf{H}_{\boldsymbol{\theta}}$ is computationally expensive. To avoid these issues, we can alternatively estimate the influence of upweighting a training node $v_\star$ on the loss for a testing node $v_{\text{test}}$:

$$\begin{aligned}\mathcal{I}_{\text{up,loss}}(v_\star, v_{\text{test}}) &\overset{\text{def}}{=} \left.\frac{\partial \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}_{\boldsymbol{\epsilon},\star}}, v_{\text{test}})}{\partial \epsilon}\right|_{\epsilon=0} \\ &= \nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_{\text{test}}) \left.\frac{\partial \boldsymbol{\theta}_{\boldsymbol{\epsilon},\star}}{\partial \epsilon}\right|_{\epsilon=0} \\ &= -\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_{\text{test}})^{\mathrm{T}}\mathbf{H}_{\boldsymbol{\theta}}^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_\star).\end{aligned} \tag{11}$$

Therefore, we can sum all testing nodes' influence $\mathcal{I}_{\text{up,loss}}(v_\star, v_{\text{test}})$ to derive the influence of the training node $v_\star$. During this process, one can use implicit Hessian-vector products (HVPs) to approximate $w_{\text{test}} \stackrel{\text{def}}{=} \mathbf{H}_{\boldsymbol{\theta}}^{-1} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_{\text{test}})^{\mathrm{T}}$ (Koh and Liang 2017), i.e., $\mathcal{I}_{\text{up,loss}}(v_\star, v_{\text{test}}) = -w_{\text{test}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_\star)$, so as to speed up the computation. Since the Hessian $\mathbf{H}_{\boldsymbol{\theta}}$ is positive semi-definite by assumption, we have:

$$w_{\text{test}} \equiv \arg\min_{\alpha} \left\{ \frac{1}{2} \alpha^{\mathrm{T}} \mathbf{H}_{\boldsymbol{\theta}} \alpha - \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(f_{\boldsymbol{\theta}}, v_{\text{test}})^{\mathrm{T}} \alpha \right\},$$

where the exact solution $\alpha$ can be obtained with conjugate gradients that only requires the evaluation of $\mathbf{H}_{\boldsymbol{\theta}} \alpha$ instead of explicitly computing $\mathbf{H}_{\boldsymbol{\theta}}^{-1}$.

We hypothesize that the larger the influence of $v_\star$, the more representative $v_\star$ for this task. Thus, we choose the first $e$ representative nodes in each class as our experiences. To our knowledge, we are the first to incorporate influence function into continual learning settings to guide the selection of experience samples. We also study the effectiveness of the influence function based experience selection scheme in our experiments. The empirical results verify the generalization of this strategy.

Our framework ER-GNN does not impose *any* restriction on GNNs architecture and can be easily incorporated into most of the current GNN models. In our evaluation, we implement our ER-GNN with a vanilla GAT (Veličković et al. 2018), forming an instance of our framework – ER-GAT.

## Experiments

We now present the results from the empirical evaluation of our framework for continual node classification tasks to demonstrate its effectiveness and applicability. We begin with systematically investigating to what extent the state-of-the-art GNNs forget on learning a sequence of node classification tasks, followed by the performance lift of our ER-GNN. Subsequently, we verify the applicability of our ER-GNN and study the hyperparameter sensitivity of our model.

**Datasets**: To evaluate the performance of our model on solving the CGL problem, we conduct experiments on three benchmark datasets: Cora (Sen et al. 2008), Citeseer (Sen et al. 2008), and Reddit (Hamilton, Ying, and Leskovec 2017) that are widely used for evaluating the performance of GNN models. To meet the requirements of the continual graph learning (task-incremental) setting, we construct 3 tasks on Cora and Citeseer, and each task is a 2-way node classification task, i.e., there are 2 classes in each task. For Reddit, we generate 8 tasks and each task is a 5-way node classification task due to its relatively large number of nodes and unique labels. We note that each task is a new task since classes in different tasks are entirely different. The statistics of the datasets and continual task settings are shown in Table 1.

| | Cora | Citeseer | Reddit |
|---|---|---|---|
| # Nodes | 2,708 | 3,327 | 232,965 |
| # Node Attributes | 1,433 | 3,703 | 602 |
| # Total Classes | 7 | 6 | 41 |
| # Tasks | 3 | 3 | 8 |
| # Classes in Each Task | 2 | 2 | 5 |

Table 1: Descriptive statistics and task settings of three datasets.

**Baselines**: To demonstrate the effectiveness of our proposed framework, we compare ER-GNN with the following GNNs for continual node classification tasks:

• **Deepwalk** (Perozzi, Al-Rfou, and Skiena 2014): Deepwalk uses local information from truncated random walks as input to learn a representation which encodes structural regularities.

• **Node2Vec** (Grover and Leskovec 2016): Node2Vec learns a mapping of nodes to a low-dimensional space of features that maximize the likelihood of preserving network neighborhoods of nodes.

• **GCN** (Kipf and Welling 2017): GCN uses an efficient layer-wise propagation rule based on a first-order approximation of spectral convolution on the graph.

• **GraphSAGE** (Hamilton, Ying, and Leskovec 2017): GraphSAGE learns a function that generates embeddings by sampling and aggregating features from the node's local neighborhood.

• **GAT** (Veličković et al. 2018): GAT introduces an attention-based architecture to perform node classification of graph-structured data. The idea is to compute each node's hidden representations by attending over its neighbors, following a self-attention strategy.

• **SGC** (Wu et al. 2019a): SGC reduces this complexity in GCN through successively removing nonlinearities and collapsing weight matrices between consecutive layers.

• **GIN** (Xu et al. 2019): GIN develops a simple architecture that is probably the most expressive among the class of GNNs and is as powerful as the Weisfeiler-Lehman graph isomorphism test.

**Experimental Setting**: We implement ER-GNN with GAT, forming an example of our framework – ER-GAT, together with our three experience selection strategies. In ER-GAT, we can readily obtain the embedding vector (before the last softmax layer). Thus, we get ER-GAT-MF, ER-GAT-MF$^\star$, ER-GAT-CM, ER-GAT-CM$^\star$, and ER-GAT-IM, where -MF, -MF$^\star$, -CM, -CM$^\star$, and -IM represent the mean of the attributes, the mean of embeddings, attribute space coverage maximization, embedding space coverage maximization, and influence maximization, respectively. The settings of all baselines and the network architecture (i.e., GAT) in our implementation are the same as suggested in the respective original papers. Without otherwise specified, we set the number of experiences stored in the experience buffer from each class as 1 (i.e., $e = 1$). However, we note that a larger value of $e$ would result in better performance.

**Metric**: To measure the performance in the continual graph learning setup, we use performance mean (PM) and forgetting mean (FM) as the evaluation metrics (Chaudhry et al. 2018). Taking the Cora dataset as an example, when learning 3 tasks sequentially, there are 3 accuracy values, i.e., one for each task after learning this task, and 3 forgetting values, i.e., the difference between the performance after learning a

| Datasets | Cora | | Citeseer | | Reddit | |
|---|---|---|---|---|---|---|
| Methods \ Metrics | PM | FM | PM | FM | PM | FM |
| DeepWalk | 85.63% | 34.51% | 64.79% | 25.92% | 76.93% | 33.24% |
| Node2Vec | 85.99% | 35.46% | 65.18% | 24.87% | 78.24% | 34.66% |
| GraphSAGE | 94.15% | 37.73% | 81.26% | 28.06% | 95.01% | 40.06% |
| GIN | 90.17% | 33.81% | 74.92% | 27.42% | 93.75% | 36.28% |
| GCN | 93.62% | 31.90% | 80.63% | 25.47% | 94.43% | 35.17% |
| SGC | 93.06% | 33.93% | 78.18% | 28.31% | 94.01% | 38.59% |
| GAT | 94.19% | 30.84% | 81.37% | 25.06% | 95.13% | 34.97% |
| ER-GAT-Random | 93.58% | 29.17% | 81.48% | 23.73% | 93.84% | 32.79% |
| ER-GAT-MF | 94.15% | 22.49% | 80.03% | 17.96% | 94.18% | 26.44% |
| ER-GAT-MF$^\star$ | 94.23% | 21.88% | **81.83%** | 17.83% | 94.63% | 23.54% |
| ER-GAT-CM | 93.98% | 22.14% | 78.78% | 18.03% | 93.33% | 26.17% |
| ER-GAT-CM$^\star$ | 94.25% | **21.03%** | 80.86% | 17.86% | 94.23% | 23.15% |
| ER-GAT-IM | **95.66%** | 21.14% | 80.85% | **17.08%** | **95.36%** | **23.09%** |

Table 2: Performance comparisons among algorithms. The bold values denote the best performance.

particular task and the performance after learning its subsequent tasks. The evaluation on Citeseer is the same as Cora. However, we use Micro F1 score as the performance metric instead of accuracy due to the imbalance between the nodes in different classes in the Reddit dataset. That is, PM and FM in Reddit represent the average Micro F1 Score and the average difference in Micro F1 Score, respectively.

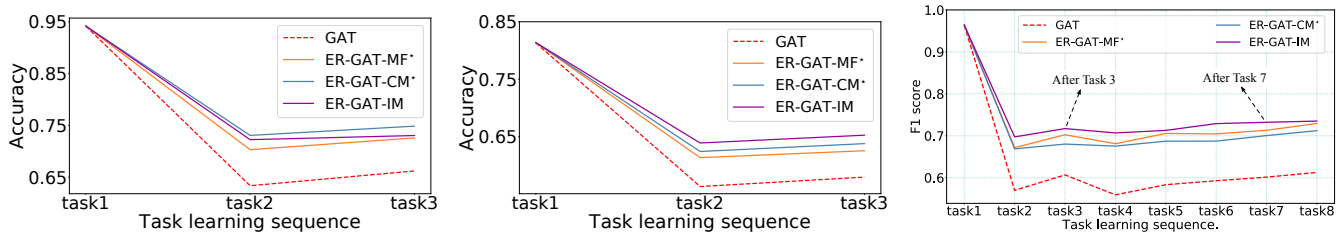
**Catastrophic Forgetting in GNNs.** We first systematically evaluate the extent of catastrophic forgetting in current GNNs. Table 2 shows the results of performance comparison on node classification, from which we can observe that all GNNs as well as two graph embedding models suffer from catastrophic forgetting problems on previous tasks. For example, the FM value on Cora, Citeseer, and Reddit are 30+%, 24+%, and 32+%, respectively. Interestingly, in some cases (e.g., Reddit), DeepWalk and Node2Vec perform better than GNNs in terms of FM, although their classification performance (i.e., PM) is lower compared to the GNNs. DeepWalk and Node2vec use truncated random walks for sampling node sequences. Once the number of sampled node sequences is large enough, the obtained node embeddings are more robust for continuous task learning, although their performance is not comparable to others on single-task learning. Therefore, it seems DeepWalk (or Node2Vec) sacrifices the performance on learning new tasks (i.e., plasticity) to relieve the catastrophic forgetting (i.e., stability) issue.

We also find some impressive results by examining the performance of different GNNs. First, as a simplified GCN, SGC is significantly faster than other GNN models because it removes the nonlinearity between the layers in GNN – it only keeps the final one. We speculate that nonlinearity in-between multi-layers in GNN can help the model remember knowledge from previous tasks. Besides, GraphSAGE achieves higher PM, but it is prone to previous task forgetting, mainly because it uses pooling operation (e.g., mean here) for feature aggregation, which, however, may smooth unique node features in previous tasks. GIN is mediocre on Cora and Citeseer but performs well on Reddit. The reason for that is because GIN may suffer from overfitting issues on relatively small datasets, as observed in (Wu et al. 2019a). In contrast, GAT performs well in terms of both PM and FM, which suggests that the attention mechanism is beneficial for both learning a new task and to a certain extent resisting catastrophic forgetting in continual graph-related task learning.

**Performance of ER-GNN.** Next, we compare our framework ER-GNN against other graph learning baselines. In addition to the three proposed node selection strategies, we also implement a random node selection scheme, called ER-GAT-Random, which randomly selects the experience nodes. Note that we report the average results of 10 runs. Table 2 shows the results of 3 different experience selection strategies. Our framework successfully decreases the FM values by a significant margin without losing the ability to learn new tasks. The IM strategy performs favorably, which proves our motivation to exploit influence function for experience replay. Another expected finding is that ER-GAT-MF$^\star$ and ER-GAT-CM$^\star$ consistently outperform ER-GAT-MF and ER-GAT-CM, which indicates the more discriminative representations in the embedding space than the attribute space.

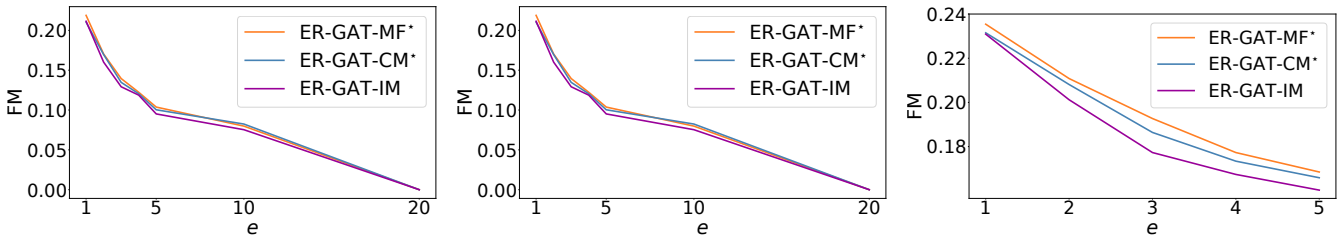
An important observation is that our ER-GAT performs comparably with vanilla GAT in terms of PM, and in some cases, our ER-GAT even outperforms the original GAT. This result implies that our approach does not sacrifice the plasticity since we expand the training set by augmenting the experience buffer nodes from previous tasks. This property is appealing as it resembles our human intelligence, i.e., we humans can not only remember previous tasks but also exploit the knowledge from preceding tasks to facilitate learning future tasks.

To provide further insight into our approach, we plot the performance evolution of the models along with the increased tasks on three benchmarks in Figure 2. For clarity, we only plot GAT's results as it performs best among baselines, with the same reason for omitting ER-GAT-MF and ER-GAT-CM. From Figure 2, we can observe that our




(a) Cora (b) Citeseer (c) Reddit

Figure 2: Performance of PM evolved with the tasks.




(a) Cora (b) Citeseer (c) Reddit

Figure 3: The influence of $e$.

framework alleviates forgetting by a large margin compared with the original GAT.

**Influence of $e$.** The number of nodes stored in the buffer from each class is among the most crucial hyperparameters. We investigate its effect and present the results in Figure 3. As expected, the more nodes stored in the buffer, the better the performance on alleviating the catastrophic forgetting. Furthermore, we can see that our model would not forget if we keep all training nodes in previous tasks – note that there are 20 training nodes in each class on Cora and Citeseer dataset.

| Model \ Datasets | Cora | Citeseer | Reddit |
|---|---|---|---|
| ER-SGC-MF | **25.00%** | 20.34% | 27.62% |
| ER-SGC-CM | 25.46% | 19.38% | 26.27% |
| ER-SGC-IM | 26.11% | **17.16%** | **25.04%** |
| ER-GIN-MF | 27.98% | 21.01% | 25.36% |
| ER-GIN-CM | 27.38% | 20.72% | 24.23% |
| ER-GIN-IM | **26.74%** | **20.68%** | **23.75%** |

Table 3: Forgetting mean of ER-SGC and ER-GIN.

**Applicability of ER-GNN.** To demonstrate the applicability of our method in other GNNs, we instantiate our experience selection schemes with SGC and GIN, forming several variants of ER-SGC and ER-GIN. As shown in Table 3, ER-SGC and ER-GIN reduce the degree of forgetting in SGC and GIN by a large margin, which demonstrates the applicability of our approaches. Besides, we also observe that the IM-based models usually achieves the lowest FM results.

## Conclusion

In this work, we formulated a novel practical graph-based continual learning problem, where the GNN model is expected to learn a sequence of node classification tasks without catastrophic forgetting. We presented a general framework called ER-GNN that exploits experience replay based methods to mitigate the impacts of forgetting. We also discussed three experience selection schemes, including a novel one – IM (Influence Maximization), which utilizes influence function to select experience nodes. The extensive experiments demonstrated the effectiveness and applicability of our ER-GNN. As part of our future, we plan to extend the continual learning to different graph-related tasks, such as graph alignment and cascade/diffusion prediction of continuously evolving graphs.

## Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No.62072077 and No.61602097).

## Broader Impact

The methods described in this paper can potentially be harnessed to improve accuracy in any application of graph neural networks where it is more expensive or difficult to retrain the models frequently. For example, the graph-based recommender systems should make recommendations to a large number of new users. However, it is often prohibitive to continually update the online model to provide the most up-to-date recommendations. Our method offers a cost-efficient solution for such incremental recommender systems. In this spirit, this work may be beneficial to a range of applications requiring stable model performance but subjecting to limited

computational resources. Although we anticipate a positive impact for applications where model predictions' robustness and stability are essential, e.g., in E-commerce and intelligent traffic systems, we also recognize possible malicious applications such as unauthorized surveillance and privacy intrusion.

## References


Chaudhry, A.; Dokania, P. K.; Ajanthan, T.; and Torr, P. H. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *ECCV*.

Chen, C.; Li, K.; Teo, S. G.; Zou, X.; Wang, K.; Wang, J.; and Zeng, Z. 2019. Gated residual recurrent graph neural networks for traffic prediction. In *AAAI*, volume 33, 485–492.

Chen, Z.; and Liu, B. 2016. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 10(3): 1–145.

de Bruin, T.; Kober, J.; Tuyls, K.; and Babuška, R. 2016. Improved deep reinforcement learning for robotics through distribution-based experience retention. In *IROS*.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2019. Continual learning: A comparative study on how to defy forgetting in classification tasks. *arXiv:1909.08383* .

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

Gao, H.; Wang, Z.; and Ji, S. 2018. Large-scale learnable graph convolutional networks. In *SIGKDD*.

Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv:1312.6211* .

Grossberg, S. T. 2012. *Studies of mind and brain: Neural principles of learning, perception, development, cognition, and motor control*, volume 70. Springer Science & Business Media.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*.

Hampel, F. R.; Ronchetti, E. M.; Rousseeuw, P. J.; and Stahel, W. A. 2011. *Robust statistics: the approach based on influence functions*, volume 196. John Wiley & Sons.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* .

Ingraham, J.; Garg, V.; Barzilay, R.; and Jaakkola, T. 2019. Generative models for graph-based protein design. In *NeurIPS*, 15820–15831.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. 2017. Overcoming catastrophic forgetting in neural networks. *PNAS* 114(13): 3521–3526.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *ICML*.

Kumaran, D.; Hassabis, D.; and McClelland, J. L. 2016. What learning systems do intelligent agents need? Complementary learning systems theory updated. *Trends in cognitive sciences* 20(7): 512–534.

Li, Z.; and Hoiem, D. 2017. Learning without forgetting. *TPAMI* 40(12): 2935–2947.

Lopez-Paz, D.; and Ranzato, M. 2017. Gradient episodic memory for continual learning. In *NIPS*.

Mallya, A.; Davis, D.; and Lazebnik, S. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *ECCV*, 67–82.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*.

McClelland, J. L.; McNaughton, B. L.; and O'Reilly, R. C. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* 102(3): 419.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *CVPR*.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine* 29(3): 93–93.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In *WWW*, 1067–1077.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. In *ICLR*.

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019. Heterogeneous graph attention network. In *WWW*, 2022–2032.

Wu, F.; Zhang, T.; Souza Jr, A. H. d.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019a. Simplifying graph convolutional networks. In *ICML*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Yu, P. S. 2019b. A comprehensive survey on graph neural networks. *arXiv:1901.00596* .

Xu, J.; and Zhu, Z. 2018. Reinforced continual learning. In *NeurIPS*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *ICLR*.

Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W. L.; and Leskovec, J. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*, 974–983.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual learning through synaptic intelligence. In *ICML*.

Zhang, C.; Song, D.; Huang, C.; Swami, A.; and Chawla, N. V. 2019. Heterogeneous graph neural network. In *SIGKDD*.