# INCREMENTAL LEARNING ON GROWING GRAPHS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graphs have attracted numerous attention in varied areas and are dynamic in many scenarios. Among dynamic graphs, growing graphs with frequently expanding vertex and edge sets are typical and widely existed, *e.g.* the rapidly growing social networks. Confronting such growing data, existing methods on either static or dynamic graphs take the entire graph as a whole and may suffer from high computation cost and memory usage due to the continual growth of graphs. To tackle this problem, we introduce incremental graph learning (IGL), a general framework to formulate the learning on growing graphs in an incremental manner, where traditional graph learning method could be deployed as a basic model. We first analyze the problems of directly finetuning on the incremental part of graph, and theoretically discuss the unbiased and edge-preserved conditions of IGL. In our method, when the graph grows with new-coming data, we select or generate vertices and edges within restricted sizes from the previous graph to update current model together with the new data. Here, two strategies, *i.e.* sample-based and cluster-based, are proposed for learning with restricted time and space complexity. We conduct experiments on the node classification and link prediction tasks of multiple datasets. Experimental results and comparisons show that our method achieves satisfying performance with high efficiency on growing graphs.

## 1 INTRODUCTION

Graph represents complicated relations of data in non-Euclidean domain with topological structure. Recently, graph neural networks have attracted considerable efforts and been successfully applied to various fields (Zhang et al., 2020). Most existing research concentrate on static graphs, while in real-world applications, graphs are generally dynamic in nature. The dynamics of graph include the addition and removal of vertices and edges, and the change of vertex attributes and edge weights (Harary & Gupta, 1997). Recent works for dynamic graph learning (Pareja et al., 2020; Manessi et al., 2020; Xu et al., 2020) consume dynamic structures using recurrent architectures and capture temporal patterns. However, there are less efforts concentrating on the growing graph, a typical and widely existed dynamic graphs, with continually added vertices and edges, *e.g.* social networks and citation networks. It is a challenging problem to frequently updating models for adapting the growth of graph with high efficiency, which has not been fully concerned in the literature.

Given a growing graph, we consider the incremental parts as subgraphs of the entire graph, with edges correspondingly split into intra-edges inside subgraphs and inter-edges among subgraphs, as shown in Figure 1 (a). For such data, the manner of incremental learning is an efficient strategy, where we expect to update model based on new subgraphs and consistently perform on the entire graph observed so far. Compared with incremental learning on data without inter-sample connections (*e.g.* images), incremental learning on growing graphs shows the differences that: (i) the previous vertices and edges must be stored in the database throughout the time, and may be related with new subgraphs, (ii) using previous data for updating with new data is essential, considering the utilization of inter-edges among subgraphs. To perform on the entire observed graph at each time, directly learning on the whole set is a feasible solution but time-consuming due to the continual growth of the graph. In contrast, simply learning on the new subgraph requires low complexity in time and space. But such a learning strategy introduces bias to the later-coming subgraphs and drops the inter-edges among subgraphs, which seriously corrupt the effectiveness of graph learning.

**Present work.** We present incremental graph learning (IGL) to formulate the incremental learning on growing graphs and explore efficient learning strategy, as shown in Figure 1 (b). Given a growing

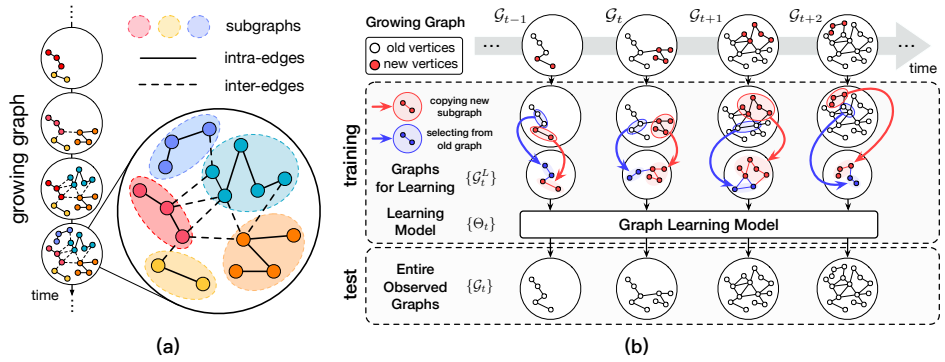


Figure 1: (a) A growing graph with sequentially added subgraphs. (b) Framework of IGL. At each time, we train on the new subgraph together with selected or generated vertices and edges from the old graph within restricted size, and aim to perform on the entire observed graph.

graph $\{\mathcal{G}_t\}$ in training, we generate a graph $\mathcal{G}_t^L$ for updating current model, where the model is modulized as a backbone and can be implemented by different graph learning methods. Then the model is expected to perform on the entire observed graph in test. To alleviate the problems of directly learning on the new subgraphs, we select or generate vertices and edges within restricted sizes from the old graph, and combine them with current subgraph as $\mathcal{G}_t^L$. Intuitively, $\mathcal{G}_t^L$ should be unbiased to the entire graph and preserve enough inter-edges. We then theoretically analyze the unbiased and edge-preserved conditions for such process, and propose sample-based and cluster-based strategies to generate $\mathcal{G}_t^L$ under specified memory constraints. Experiments of node classification and link prediction on multiple datasets are conducted for evaluations.

**Comparison with related works.** We review related works and present their main differences with IGL in motivation, target and method designing. (a) *Incremental learning* continually extends its learned knowledge based on new data, where the incremental data could be new samples, categories, domains, *etc.* Related works on data with independent samples (*e.g.* images), mainly focus on incremental categories (Rebuffi et al., 2017) or incremental domains (Li & Hoiem, 2017) to solve the problem of catastrophic forgetting (McCloskey & Cohen, 1989). Sample-incremental learning is mostly concerned in traditional machine learning methods and could be naturally conducted by neural networks with mini-batch training. Though IGL is a sample-incremental learning, the inter-edges among vertices from different time, *i.e.* independent samples, make it a challenging problem for existing methods. (b) *Inductive graph learning* could be generalized to graph structures different from the learned one under the same distribution (Hamilton et al., 2017; Zeng et al., 2019), which is suitable for dynamic graphs to some extent. However, it has to consume the entire graph at each time to capture the increasing patterns on growing graphs, and could be adapted as backbones in our IGL framework. (c) *Mini-batch based graph learning* learns on a large graph by cutting batches of subgraphs from it with mini-batch training (Chiang et al., 2019). Though also sequentially learning on subgraphs, the entire graph here keeps accessible for selecting subgraphs that could be trained multiple times. While in IGL, the split and permutation of subgraphs are specifically given as input.

**Contributions.** (i) We present IGL, a general framework to address the problem of efficiently learning on growing graphs in an incremental manner. (ii) We theoretically analyze problems of baseline solutions, and propose sample-based and cluster-based strategies to generate graphs for learning within restricted size. (iii) We conduct experiments on multiple datasets in the IGL scenario, and experimental results show satisfying performance and high efficiency of proposed methods.

## 2 Problem Definition

Incremental graph learning (IGL) is defined on a growing graph $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1)$, $\mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2)$, ..., $\mathcal{G}_t(\mathcal{V}_t, \mathcal{E}_t)$, ..., where $\mathcal{G}_t(\mathcal{V}_t, \mathcal{E}_t)$ indicates the graph at time $t$ with the vertex set $\mathcal{V}_t$ and edge set $\mathcal{E}_t$. Edge $(\mathbf{v}_i, \mathbf{v}_j)$ in $\mathcal{E}_t$ represents an undirected connection between $\mathbf{v}_i$ and $\mathbf{v}_j$ in $\mathcal{V}_t$. The growth of graph suggests $\forall i < j, \mathcal{V}_i \subseteq \mathcal{V}_j, \mathcal{E}_i \subseteq \mathcal{E}_j$. The set of new vertices and edges at time $t$ are noted as $\mathcal{V}_t^{new} = \mathcal{V}_t - \mathcal{V}_{t-1}$, $\mathcal{E}_t^{new} = \mathcal{E}_t - \mathcal{E}_{t-1}$. And $\mathcal{E}_t^{new}$ could be split into two subsets: (i) intra-edges $\mathcal{E}_t^{intra}$ among vertices in $\mathcal{V}_t^{new}$, (ii) inter-edges $\mathcal{E}_t^{inter}$ between vertices in $\mathcal{V}_t^{new}$ and $\mathcal{V}_{t-1}$.

Considering any element-level graph learning task $\mathcal{T}$, *e.g.* node classification, with $\Theta$ as parameters of learning model, we define $f_{\mathcal{T}}(\mathcal{G}, \Theta)$ for measuring $\Theta$'s performance on $\mathcal{G}$. Then the optimization

**Algorithm 1** Framework of incremental graph learning.

**Input:** $\mathcal{G}_1(\mathcal{V}_1, \mathcal{E}_1), \mathcal{G}_2(\mathcal{V}_2, \mathcal{E}_2), ..., \mathcal{G}_t(\mathcal{V}_t, \mathcal{E}_t), ...$ // the growing graph
**Input:** $V_{max}$, $E_{max}$ // restrictions to additional size of learned graph
Initialize learning parameters as $\Theta_0$
**for** $t = 1, 2, ...$ **do**
$\mathcal{V}_t^{new} \leftarrow \mathcal{V}_t - \mathcal{V}_{t-1}$, $\mathcal{E}_t^{new} \leftarrow \mathcal{E}_t - \mathcal{E}_{t-1}$
$\mathcal{E}_t^{intra} \leftarrow \{(\mathbf{v}_i, \mathbf{v}_j) \in \mathcal{E}_t^{new} \mid \mathbf{v}_i, \mathbf{v}_j \in \mathcal{V}_t^{new}\}$, $\mathcal{E}_t^{inter} \leftarrow \mathcal{E}_t^{new} - \mathcal{E}_t^{intra}$
Generate additional vertex set $\Delta\mathcal{V}_t^L$ and edge set $\Delta\mathcal{E}_t^L$, *s.t.* $|\Delta\mathcal{V}_t^L| \leq V_{max}$, $|\Delta\mathcal{E}_t^L| \leq E_{max}$
$\mathcal{V}_t^L \leftarrow \mathcal{V}_t^{new} \cup \Delta\mathcal{V}_t^L$, $\mathcal{E}_t^L \leftarrow \mathcal{E}_t^{intra} \cup \Delta\mathcal{E}_t^L$
Update parameters $\Theta_t \leftarrow GraphLearning(\mathcal{G}_t^L(\mathcal{V}_t^L, \mathcal{E}_t^L), \Theta_{t-1})$
**end for**

target of IGL at time $t$ is $\Theta_t = \arg\max_\Theta f_\mathcal{T}(\mathcal{G}_t, \Theta)$. To make our framework capable of being generalized to different graph learning tasks and adapting existing methods, we suppose to generate a graph $\mathcal{G}_t^L(\mathcal{V}_t^L, \mathcal{E}_t^L)$ for optimizing $\Theta$ with a static graph learning method. The method could be regarded as a backbone of the framework. A straightforward solution, named as *joint training*, is to directly optimize on the entire graph, *i.e.* $\mathcal{G}_t^L = \mathcal{G}_t$. Due to the growth of graph, joint training suffers from increasing computation cost and memory usage. Another lightweight choice, *finetuning*, is to learn on the new subgraph, *i.e.* $\mathcal{G}_t^L = \mathcal{G}_t^{new}(\mathcal{V}_t^{new}, \mathcal{E}_t^{intra})$ and drop the inter edges $\mathcal{E}_t^{inter}$, but dropping edges may seriously corrupt the effectiveness of learning (Dai et al., 2018). To utilize $\mathcal{E}_t^{inter}$ with efficient computation, we allow to assign additional memory to expand the vertex and edge sets of $\mathcal{G}_t^{new}$ within restricted sizes. Based on the above ideas, the framework of IGL is formulated in Algorithm 1. It is noted that we modulize the backbone learning method as the function $GraphLearning(\mathcal{G}, \Theta)$, which learns on $\mathcal{G}$ with parameters initialized as $\Theta$ for the specified task.

We conclude properties of a qualified IGL method as follows: (i) It takes a sequence of growing graph as input. (ii) During the training stage at each time, the entire graph observed so far is freely available only in the preprocessing time. Memory size and computation complexity for learning should be restricted within specified boundaries, or increase relatively slow compared with the growth of graph. (iii) It performs effectively on the entire graph observed so far in test.

# 3 METHOD

## 3.1 THEORETICAL MOTIVATIONS

Suppose the $GraphLearning()$ process in IGL is performed by the graph neural networks consisting of aggregators and updaters (Zhou et al., 2018). Since finetuning learns on each new subgraph *s.t.* $\mathcal{G}_t^L = \mathcal{G}_t^{new}$, it intuitively introduces the following problems: (i) **Subgraph bias.** Since the propagation of graph learning is conducted inside each subgraph, the parameters for learning aggregating results tend to the bias of subgraphs rather than the entire graph. (ii) **Inter-edges missing.** The inter connections $\mathcal{E}_t^{inter}$ among different subgraphs are completely dropped during the training, which blocks the aggregation process through edges and disturbs the effectiveness of graph learning. To address these problems, following conditions should be considered for $\mathcal{G}_t^L(\mathcal{V}_t^L, \mathcal{E}_t^L)$.

**Unbiased Estimation** To alleviate the bias of subgraph, the aggregation results of vertices in $\mathcal{V}_t^L$ for learning should be unbiased estimations of them in the entire graph:

$$\forall \mathbf{v} \in \mathcal{V}_t,\ \mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L\right) = agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right), \tag{1}$$

where $\mathcal{N}_t(\mathbf{v}) = \{\mathbf{u} \in \mathcal{V}_t \mid (\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t\}$, and $agg(\mathbf{v}, \mathcal{N})$ aggregates embeddings from vertices in $\mathcal{N}$ to $\mathbf{v}$, details of which may vary in different methods. For convenience of further analysis, we consider the mean aggregator (Hamilton et al., 2017) within one layer, and assume that all related edges to $\mathcal{V}_t^L$ are kept in $\mathcal{E}_t^L$. Then we show that uniform sampling in $\mathcal{V}_t$ satisfies the condition (1):

**Theorem 4.1** *Suppose* $agg(\mathbf{v}, \mathcal{N}) = \Sigma_{\mathbf{u} \in \mathcal{N}} \frac{\mathbf{u}}{|\mathcal{N}|}$, *and* $\mathcal{E}_t^L = \{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t \mid \mathbf{u}, \mathbf{v} \in \mathcal{V}_t^L\}$. *If* $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_t, P(\mathbf{u}_1 \in \mathcal{V}_t^L) = P(\mathbf{u}_2 \in \mathcal{V}_t^L)$, *then for any possible size* $|\mathcal{V}_t^L|$, *equation (1) holds.*

Under specified memory limits, however, keeping equal probability of sampling in $\mathcal{V}_t$ will inevitably reduce the utilization of later-coming subgraphs $\mathcal{G}_t^{new}$ to be $O(1/t)$, and restricts the graph learning efforts. Thus, we relax the above condition to be **soft-unbiased** that holds all the $\mathcal{V}_t^{new}$ in $\mathcal{V}_t^L$ and conducts uniform sampling in $\mathcal{V}_{t-1}$ for the additional vertex set $\Delta\mathcal{V}_t^L = \mathcal{V}_t - \mathcal{V}_t^L$.

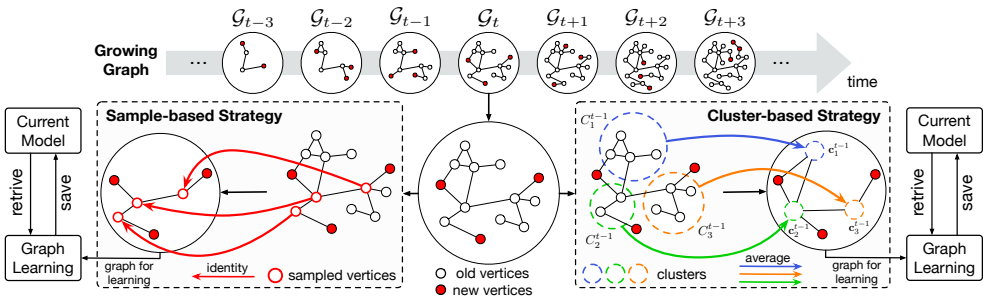


Figure 2: **Pipeline of our proposed methods for IGL**, where we can choose the sample-based or cluster-based strategy, generating a graph for learning to update current model at each time.

**Theorem 4.2** *If* $\forall \mathbf{u} \in \mathcal{V}_t^{new}, \mathbf{u} \in \mathcal{V}_t^L$, *and* $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_{t-1}, P(\mathbf{u}_1 \in \mathcal{V}_t^L) = P(\mathbf{u}_2 \in \mathcal{V}_t^L)$, *then for any possible size* $|\mathcal{V}_t^L|$, $\forall \mathbf{v} \in \mathcal{V}_t, \mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L\right)$ *equals*

$$\mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|}\right) agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right) - \mathbb{E}\left(\frac{|\mathcal{N}_{t-1}\left(\mathbf{v}\right) - \mathcal{V}_t^L|}{|\mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L|}\right) agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v})). \tag{2}$$

Let $\lambda_1 = \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|}\right)$, $\lambda_2 = -\mathbb{E}\left(\frac{|\mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|}\right)$, we could get $\lambda_1 + \lambda_2 = 1$ and the expectation of aggregation result as $\lambda_1 agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right) + \lambda_2 agg\left(\mathbf{v}, \mathcal{N}_{t-1}\left(\mathbf{v}\right)\right)$, a weighted average of the full aggregation in $\mathcal{G}_t$ and $\mathcal{G}_{t-1}$. Since the initial parameter $\Theta_{t-1}$ has been updated for $agg\left(\mathbf{v}, \mathcal{N}_{t-1}\left(\mathbf{v}\right)\right)$ and $\lambda_1 \geq 1, \lambda_2 \leq 0$, the expectation of current aggregation provides an unbiased optimizing direction towards target $agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right)$. And $\lambda_1 \to 1, \lambda_2 \to 0$ when $V_{max}$ increases. Thus, we present an soft-unbiased estimation with full utilization of new subgraphs.

**Edge Preservation** Since the missing of inter-edges may seriously affect the graph learning results, we aims at preserving more edges of $\mathcal{E}_t^{inter}$ in $\Delta\mathcal{E}_t^L$, which could be formulated as

$$\max_{\Delta\mathcal{E}_t^L} |\Delta\mathcal{E}_t^L \cap \mathcal{E}_t^{inter}|, \ s.t. \ |\Delta\mathcal{E}_t^L| \leq E_{max}. \tag{3}$$

The edge-preservation could be required as a definite optimization problem in (3), or sampling with priority to vertices with higher degrees so that $P(\mathbf{u} \in \mathcal{V}_t^L) \propto |\{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter} \mid \mathbf{v} \in \mathcal{V}_t^{new}\}|$, which naturally shows conflict with the unbiased condition in practice.

Our proposed methods basically follow the unbiased and edge-preserved conditions. We first present the sample-based strategy (3.2) that selectively satisfies part of the conditions, and the following cluster-based strategy (3.3) presents a cluster-graph that mediately satisfies both of them.

### 3.2 Sample-based Strategy

We consider sampling a subset $\Delta\mathcal{V}_t^L$ from $\mathcal{V}_{t-1}$ within size of $V_{max}$, and preserving all the related edges, *i.e.* $E_{max} = (|\mathcal{V}_t^{new}| + V_{max})^2 - |\mathcal{E}_t^{intra}|$ by default.

$$\Delta\mathcal{V}_t^L = SampleVertices(\mathcal{G}_{t-1}, V_{max}), \ \ \Delta\mathcal{E}_t^L = \{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter} \mid \mathbf{u} \in \Delta\mathcal{V}_t^L, \mathbf{v} \in \mathcal{V}_t^{new}\}, \tag{4}$$

where the sampling function $SampleVertices()$ has been widely studied in the literature for selecting a representative subgraph from the original graph (Hu & Lau, 2013). Considering the above required conditions, we explore following pragmatic methods for appropriate sampling.

**Random Selection** uniformly selects $V_{max}$ vertices from $\mathcal{V}_{t-1}$, which absolutely follows the unbiased condition, but ignores to preserve enough edges for learning, especially in sparse graphs.

**Random Jump** is a traversal-based sampling (Leskovec & Faloutsos, 2006) and we adapt it in the following steps: Starting with any vertex in $\mathcal{V}_t^{new}$, we either randomly walk to a neighboring vertex in $\mathcal{V}_{t-1}$ with probability $p$ and select it, or randomly jump to a vertex in $\mathcal{V}_t^{new}$ with probability $(1-p)$. Repeat to fill the sampled set. Zhao et al. (2019) show that the probability of sampling a vertex tends to be proportional to its degree, which partly meets the edge-preserved condition.

**Degree-based Selection** is inspired directly from the edge-preserved condition to sample with priority to vertices related to more inter-edges. Let $D_t(\mathbf{u}) = |\{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t\}|$ be degree of $\mathbf{u}$, we define $D_t^{new}(\mathbf{u}) = \frac{D_t(\mathbf{u}) - D_{t-1}(\mathbf{u})}{D_t(\mathbf{u})}, \forall \mathbf{u} \in \mathcal{V}_{t-1}$ as **new degree** of vertices to reflect their closeness to the new subgraph through inter-edges. Then we select top-$V_{max}$ vertices in $\mathcal{V}_{t-1}$ by their new degrees.

The above methods consider only part of required conditions. It can be proved that, ignoring the ideal case when all the vertices in $\mathcal{V}_{t-1}$ connect with same number of vertices in $\mathcal{V}_t^{new}$, such sampling in (4) satisfies the two required conditions *iff* all the vertices are sampled, *i.e.* joint training.

**Theorem 4.3** *Suppose* $\exists \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_{t-1}$, *s.t.* $D_t(\mathbf{u}_1) - D_{t-1}(\mathbf{u}_1) \neq D_t(\mathbf{u}_2) - D_{t-1}(\mathbf{u}_2)$. *Then the unbiased and edge-preserved conditions hold iff* $\forall \mathbf{u} \in \mathcal{V}_{t-1}, P(\mathbf{u} \in \Delta\mathcal{V}_t^L) = 1$, *i.e.* $\mathcal{V}_t^L = \mathcal{V}_t$.

### 3.3 Cluster-based Strategy

Theorem 4.3 suggests that the sample-based strategy could not require both unbiased estimation and edge preservation. In this section, we relax the basic assumption in sample-based strategy that $\mathcal{G}_t^L$ should be a subgraph of $\mathcal{G}_t$, and construct a **cluster-graph** to mediately satisfies both conditions. Suppose vertices in $\mathcal{V}_{t-1}$ are arranged into $K$ cluster sets $\{\mathcal{C}_i^{t-1}\}_{i=1}^K$ with centers $\{\mathbf{c}_i^{t-1}\}_{i=1}^K$, where $\mathbf{c}_i^{t-1} = \frac{1}{|\mathcal{C}_t^{t-1}|}\Sigma_{\mathbf{v}\in\mathcal{C}_t^{t-1}}\mathbf{v}$. Then the cluster-graph is defined as

$$\begin{cases} \Delta\mathcal{V}_t^L = \{\mathbf{c}_1^{t-1}, ..., \mathbf{c}_K^{t-1}\} \\ \Delta\mathcal{E}_t^L = \{(\mathbf{c}_i^{t-1}, \mathbf{v}) \mid \mathbf{v} \in \mathcal{V}_t^{new}, \exists\, \mathbf{u} \in \mathcal{C}_i^{t-1}, (\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter}\} \cup \\ \qquad\qquad \{(\mathbf{c}_i^{t-1}, \mathbf{c}_j^{t-1}) \mid \exists\, \mathbf{u}_1 \in \mathcal{C}_i^{t-1}, \mathbf{u}_2 \in \mathcal{C}_j^{t-1}, (\mathbf{u}_1, \mathbf{u}_2) \in \mathcal{E}_{t-1}\} \end{cases}, \tag{5}$$

which suggests that the cluster centers are added as new cluster-vertices, and the edges connecting to any vertex in $\mathcal{V}_{t-1}$ are directly transferred to the corresponding cluster-vertex. It is noted that the additional edge sets in equation (5) represent $\mathcal{E}_t^{inter}$ and $\mathcal{E}_{t-1}$, respectively.

In the cluster-graph, $\mathcal{E}_t^{inter}$are approximately preserved by connecting cluster-vertices. For the unbiased condition, here $\mathcal{V}_{t-1}$ are not directly included in $\mathcal{V}_t^L$. Let $i(\mathbf{u}) = k$ if $\mathbf{u} \in \mathcal{C}_k^{t-1}$, we could replace $P(\mathbf{u} \in \mathcal{V}_t^L)$ for calculating the expectation of aggregation by $\frac{1}{|\mathcal{C}_{i(\mathbf{u})}^{t-1}|}P(\mathbf{c}_{i(\mathbf{u})}^{t-1} \in \mathcal{V}_t^L) = \frac{1}{|\mathcal{C}_{i(\mathbf{u})}^{t-1}|}$, with an error proportional to the tightness of clustering. Thus, the cluster-vertices tend to uniform sampling from $\mathcal{V}_{t-1}$ when the clusters are within the same size, *i.e.* balanced clustering.

Due to the continual growth of graph, direct clustering on the entire graph is time-consuming. For an approximate but efficient clustering with balanced size, we first conduct clustering on the new vertices $\mathcal{V}_t^{new}$ for cluster sets $\{\Delta\mathcal{C}_i^t\}_{i=1}^K$ and corresponding centers $\{\hat{\mathbf{c}}_i^t\}_{i=1}^K$. Bipartite matching algorithm is applied to optimize a bijective marching function $M(\cdot) : \{1, ..., K\} \to \{1, ..., K\}$ for the objective: $\min_{m(\cdot)} \Sigma_{k=1}^K \|\mathbf{c}_k^{t-1} - \hat{\mathbf{c}}_{m(k)}^t\|_2^2$, which assigns new clusters to closer old clusters. Then we merge the clusters as $\mathcal{C}_k^t = \mathcal{C}_k^{t-1} \cup \Delta\mathcal{C}_{m(k)}^t$ and update the center $\mathbf{c}_k^t$ correspondingly.

We adapt two methods for the implementation of balanced clustering: (i) **random grouping** that uniformly assigns vertices into groups as a baseline method, (ii) the **constrained k-means clustering** algorithm (Bradley et al., 2000) with the restricted minimum cluster size $\lfloor|\mathcal{V}_t^{new}|/K\rfloor$.

### 3.4 Information Theory Analysis

We present a theoretical analysis to evaluate the effectiveness of methods from the perspective of information theory. The old graph is regarded as unknown variables and we compare the quantities of information provided by different methods. Formally, old vertices are simplified as $n$ random variables $X = \{x_i\}_{i=1}^n$ in $\{0, 1\}$. We use the information entropy in *Boltzmann formula* that $S = k_B \log \Omega$, where $k_B$ is a constant and $\Omega$ is the number of *microstates* (*i.e.* possible states of all variables). Specifically, the entropy in finetuning and joint training are $k_B \log 2^n$ and $k_B \log 1 = 0$. We regard the proposed methods as $r$ constraints $e_1, ..., e_r$ to $X$, and the entropy as $S(e_1, ..., e_r)$, so the quantities of information $\Delta S = S_0 - S(e_1, ..., e_r)$ and we compare $\Delta S$ as follows.

**Theorem 4.4** *Let* $f(i)$ *denotes the equation* $x_i = v$, *and* $g(i, j)$ *denotes the equation* $\Sigma_{k=i}^j x_k = m$, *where* $v, m$ *is given by the real distribution of* $X$. *Let* $l_1, l_2, ..., l_r$ *be the size of clusters, where* $\Sigma_{i=1}^r l_i = n$ *and* $L_k = \Sigma_{i=1}^k l_i$. *Then the following inequalities holds:*

$$\begin{aligned} &S_c = S(g(1, \frac{n}{r}), g(\frac{n}{r} + 1, \frac{2n}{r}), ..., g(\frac{(r-1)n}{r} + 1, n)) \leq S(f(1), f(2), ..., f(r)), \\ &\mathbb{E}(S_c) \leq \mathbb{E}(S(g(1, L_1), g(L_1 + 1, L_2), ..., g(L_{r-1} + 1, n))). \end{aligned} \tag{6}$$

Theorem 4.4 shows that the cluster-based strategy with balanced clustering provides larger $\Delta S$, *i.e.* more information quantities. Detailed discussions and proofs are in Appendix E.4.

Table 1: **Average accuracy (%) of node classification in IGL.** The value $T$ under each dataset denotes the temporal length of growing graph. Top-3 results are marked in bold red, blue and black.

| | Method | **Cora** ($T = 10$) | **Citeseer** ($T = 20$) | **Pubmed** ($T = 20$) | **Flickr** ($T = 50$) | **Reddit** ($T = 100$) |
|---|---|---|---|---|---|---|
| Bounds | *finetuning (lower bound)* | 39.69 ± 2.19 | 46.94 ± 1.82 | 52.61 ± 4.22 | 56.26 ± 0.56 | 57.42 ± 3.07 |
| | *joint training (upper bound)* | 58.85 ± 0.76 | 56.56 ± 0.98 | 67.18 ± 0.61 | 63.45 ± 0.38 | 89.30 ± 0.27 |
| Compared IL Methods | EWC (Kirkpatrick et al., 2017) | 41.23 ± 1.51 | 46.65 ± 1.67 | 58.00 ± 3.27 | 56.30 ± 1.85 | 73.30 ± 2.23 |
| | LwF (Li & Hoiem, 2017) | 42.97 ± 2.35 | 46.33 ± 1.36 | 55.44 ± 4.57 | 56.83 ± 1.38 | 56.27 ± 1.44 |
| | iCaRL (Rebuffi et al., 2017) | 50.11 ± 2.03 | **53.93 ± 1.32** | 57.73 ± 0.97 | 57.58 ± 1.23 | **83.52 ± 0.57** |
| | TEM (Chaudhry et al., 2019b) | 48.22 ± 1.46 | 52.30 ± 1.40 | 58.41 ± 1.85 | 55.74 ± 0.78 | 74.12 ± 0.60 |
| | A-GEM (Chaudhry et al., 2019a) | 50.25 ± 1.54 | 52.52 ± 1.58 | 62.44 ± 1.68 | 54.55 ± 2.24 | 79.57 ± 0.67 |
| Ours (sample-) | sample-random | 49.96 ± 1.43 | 52.42 ± 1.79 | 61.73 ± 1.53 | 56.77 ± 0.59 | 79.40 ± 0.18 |
| | sample-random jump | 50.27 ± 1.67 | 52.23 ± 1.63 | 61.27 ± 1.74 | 56.53 ± 0.85 | 80.70 ± 1.00 |
| | sample-new degree | **54.90 ± 1.17** | 53.81 ± 1.65 | **62.75 ± 1.21** | **58.17 ± 0.32** | 78.25 ± 1.24 |
| Ours (cluster-) | cluster-random | **56.22 ± 1.10** | **53.99 ± 1.59** | **65.91 ± 1.02** | **60.78 ± 0.35** | **86.03 ± 0.09** |
| | cluster-cons.KMeans | **56.66 ± 0.91** | **54.50 ± 1.76** | **65.64 ± 1.18** | **60.70 ± 0.44** | **86.10 ± 0.09** |

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Datasets.** We evaluate the proposed methods on: (i) the citation networks Cora, Citeseer and Pubmed (Sen et al., 2008) with documents as vertices and citation relations as edges, (ii) the online image network Flickr (Zeng et al., 2019) with image vertices connected with common properties, (iii) the social network Reddit (Hamilton et al., 2017) with posts vertices connected with common users' comments. Since IGL is required to perform on any input of growing graphs, we randomly split and permute the vertices into $T$ groups to simulate the growth of graph.

**Tasks.** Two common graph learning tasks in element-level are conducted: (i) node classification in the semi-supervised manner that predicts categories of nodes with only a small set of nodes labeled for learning (Yang et al., 2016), (ii) link prediction that predicts the existence of connection between two vertices based on their learned embeddings (Kipf & Welling, 2016).

**Compared methods.** The results of finetuning and joint training are provided as approximate lower and upper bounds for reference. Though existing incremental learning (IL) methods mainly focus on incremental category or domain scenarios, and are not completely suitable for IGL, we adapt representative methods in recent years for a comprehensive comparison, including the regularization-based methods EWC (Kirkpatrick et al., 2017) and LwF (Li & Hoiem, 2017) without using previous data, and the replay-based methods iCaRL (Rebuffi et al., 2017), TEM (Chaudhry et al., 2019b) and A-GEM (Chaudhry et al., 2019a) that stores samples from previous data. In comparison, our proposed sample and cluster based strategies are noted starting with "sample-" and "cluster-".

Details of the datasets, configurations and compared methods are presented in the appendices.

## 4.2 RESULTS OF INCREMENTAL GRAPH LEARNING

**Node Classification.** For the backbone model in $GraphLearning()$ for node classification, we adapt GCN (Kipf & Welling, 2017) for smaller graphs Cora and Citeseer, and GraphSAGE (Hamilton et al., 2017) for the rest. The total time-steps of growing graph $T$ is specified for each dataset. For balanced category learning, we set the constraint $V_{max} = M\times$ number of categories, and conduct the sampling and clustering inside each category, and $M = 3$ in our main experiments. The influence of the above settings are further studied in Section 4.3, with details in Appendix A.

Table 1 lists the average of classification accuracy on the entire observed graph throughout the time. Compared with the finetuning, the IL methods in comparison achieve improved but not stable performance, since they are not specifically designed for graphs. Our proposed methods achieve satisfying improvements, where the cluster-based strategy outperforms the sample-based, and steps closer to the upper bound. Such performance is consistent with theoretical analysis on required conditions, and verifies that mediately satisfying both the unbiased and edge-preserved condition performs better. Additional experiments further explain to such performance that the cluster-based strategy gen-

Table 2: **Average AUC (%) of link prediction in IGL.** The '-' denotes that the results are unavailable due to out of memory on large graphs using joint training.

| | Method | Cora ($T = 10$) | Citeseer ($T = 20$) | Pubmed ($T = 20$) | Flickr ($T = 50$) | Reddit ($T = 100$) |
|---|---|---|---|---|---|---|
| Bounds | *finetuning (lower bound)* | 66.37 ± 0.47 | 60.41 ± 0.28 | 73.12 ± 1.02 | 57.58 ± 0.04 | 85.97 ± 2.39 |
| | *joint training (upper bound)* | 89.22 ± 0.08 | 93.55 ± 0.12 | 79.92 ± 0.05 | - | - |
| Compared IL Methods | EWC (Kirkpatrick et al., 2017) | 68.63 ± 1.63 | 64.37 ± 2.83 | 76.25 ± 2.80 | **61.55 ± 3.95** | 85.02 ± 3.73 |
| | LwF (Li & Hoiem, 2017) | 71.04 ± 0.82 | 63.73 ± 1.31 | **78.08 ± 0.81** | 57.05 ± 0.62 | **90.00 ± 1.99** |
| | iCaRL (Rebuffi et al., 2017) | **72.03 ± 0.63** | 61.68 ± 0.39 | 75.86 ± 1.26 | 57.19 ± 0.22 | 83.15 ± 3.82 |
| | TEM (Chaudhry et al., 2019b) | 68.30 ± 0.63 | 62.48 ± 0.39 | 75.46 ± 1.20 | 57.88 ± 0.22 | 86.45 ± 1.16 |
| | A-GEM (Chaudhry et al., 2019a) | 67.72 ± 0.86 | 64.32 ± 0.44 | 76.29 ± 0.97 | 56.66 ± 0.40 | 82.34 ± 5.98 |
| Ours (sample-) | sample-random | 68.12 ± 1.20 | **66.78 ± 0.46** | 74.74 ± 1.12 | 57.60 ± 0.24 | 88.45 ± 1.82 |
| | sample-random jump | 66.93 ± 2.87 | 66.43 ± 3.06 | 75.54 ± 1.46 | 58.13 ± 0.24 | 87.57 ± 1.76 |
| | sample-new degree | 70.33 ± 0.31 | 65.62 ± 1.84 | 76.08 ± 0.89 | 57.93 ± 0.18 | 87.96 ± 2.32 |
| Ours (cluster-) | cluster-random | **71.13 ± 0.44** | **71.45 ± 0.35** | **79.10 ± 0.41** | **60.16 ± 0.43** | **88.48 ± 0.61** |
| | cluster-cons.KMeans | **73.60 ± 0.36** | **68.27 ± 0.39** | **81.92 ± 0.55** | **63.51 ± 0.13** | **90.58 ± 0.61** |

erates more distinguishable feature distributions (Section 4.4) and preserves more adequate edges under specified memory restrictions (Section 4.5) for efficient graph learning.

**Link Prediction.** The basic settings are the same as node classification. We follow Kipf & Welling (2016) using the reconstruction loss for training and calculate the *Area Under Curve* (AUC) as evaluation metric. The memory constraint is set as $V_{max} = M_{all}$ without considering categories, where $M_{all} = 20$ for the citation networks, and $M_{all} = 100$ for Flickr and Reddit. The edge sets $\mathcal{E}_t^{intra}$ and $\mathcal{E}_t^{inter}$ are split into training, validation and test sets with the proportion of $8 : 1 : 1$. Details of sampling negative edges for training and other configurations are in Appendix B.

We report the average AUC in Table 2. Similar comparison of results are observed, except that the cluster-based strategy outperforms joint training on Pubmed, which is attributed to the newly generated edges incident to the cluster-vertices. It is noted that the clustering here is conducted among all vertices rather than inside each category, and raises stricter requirement to the tightness of clustering method. Thus, the difference between random grouping and constrained k-means in the cluster-based strategy gets larger compared with that in the node classification.

### 4.3 Ablation Studies

We conduct ablation studies on the basic configurations with node classification to observe the influence of these factors, and the results are shown in Figure 3: (a) **Total time-steps** $T$. With the increase of $T$ for the same size of graph, finetuning dramatically decreases, while our proposed methods drop relatively slow like joint training, showing their robustness to the temporal length of growing graphs. (b) **Memory constraint** $M$. The proposed methods show uptrend to joint training with larger available memory size, and maintain consistent comparison as the main experiments. (c) **Backbone models**. We conduct experiments on the same dataset using different backbones, including GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), GIN (Xu et al., 2018) and GraphSAGE (Hamilton et al., 2017). The results suggest that though theoretically analyzed under mean aggregator and tested with basic backbones, our proposed methods own the ability of adapting different static graph learning methods to the task of IGL.

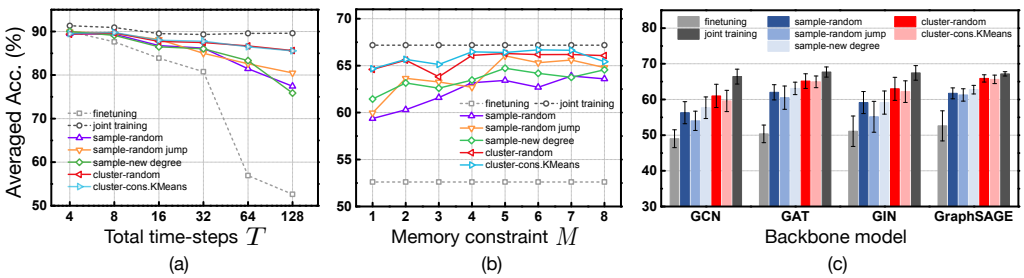


Figure 3: **Results of ablation studies in node classification.** (a) Results of Reddit with different length of growing graph sequence (in log scale). (b) Results of Reddit under different memory constraint to size of the graph for learning. (c) Results of Pubmed using different backbone models.

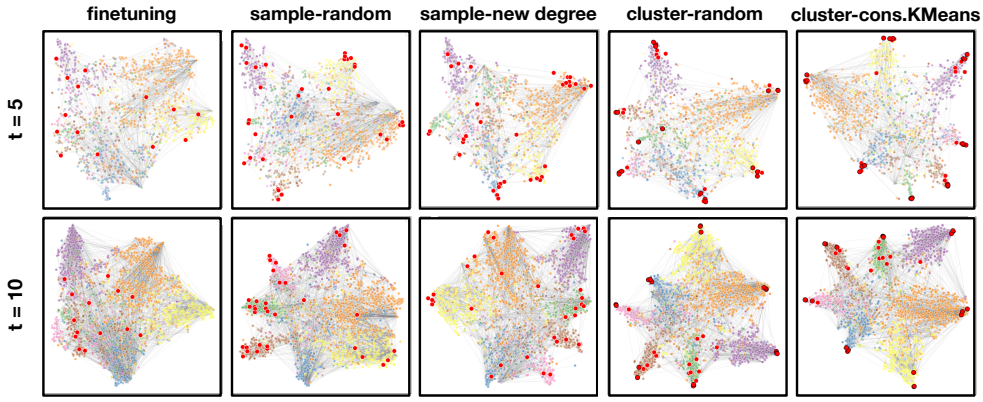


Figure 4: **Graph feature visualization using t-SNE** in node classification of Cora at time 5 and 10. Training vertices are drawn in red, where the cluster vertices are stroked in black.

### 4.4 Visualization Analysis

We visualize the learned features of vertices to analyze the effectiveness of proposed methods. In Figure 4, we apply t-SNE (Maaten & Hinton, 2008) to features of observed vertices in node classification of Cora, and color them with the label information. Features of our proposed methods show more discernible distribution of categories. To visualize the supervisions, we color the training vertices in red, with the cluster vertices stroked in black. In the cluster-based strategy, the training vertices turn to be located in those sharp places that are more representative to distinguish different categories, which could enhance the effectiveness of learning. In comparison, the distribution of training vertices are more stochastic in finetuning and the sample-based strategy.

### 4.5 Time and Space Complexity

We record the consumed time of learning, including the preprocessing and training time, to estimate the computation cost, and count the number of vertices and edges of $\mathcal{G}_t^L$ at each time as the space complexity. In comparison of proposed methods, the average results of node classification on Reddit are listed in Table 3, together with the classification accuracy for reference. Results show that it takes extremely high complexity for joint training to achieve its great performance, while our proposed methods show comparable performance and meanwhile maintain much lower complexity. And it is noted that the cluster-based strategy succeeds to preserve more inter-edges under memory restriction, which demonstrates that the edge-preserved condition is better satisfied for efficient learning.

Table 3: Average time and space complexity of node classification on Reddit.

| **Method** | *finetuning* | *joint training* | sample-random | sample-random jump | sample-new degree | cluster-random | cluster-cons.KMeans |
|---|---|---|---|---|---|---|---|
| **Time (s)** | 25.63 | 346.37 | 26.24 | 26.39 | 31.77 | 27.85 | 28.27 |
| **#Nodes (K)** | 2.33 | 117.61 | 2.57 | 2.57 | 2.57 | 2.57 | 2.57 |
| **#Edges (K)** | 11.51 | 39,282 | 14.17 | 21.18 | 12.89 | 43.53 | 43.38 |
| Acc (%) | 57.42 | 89.30 | 79.40 | 80.70 | 78.25 | 86.03 | 86.10 |

## 5 Conclusion

In this paper, we study the problem of efficient learning on growing graphs in an incremental manner, and formulate a general framework named incremental graph learning (IGL). We theoretically analyze the unbiased and edge-preserved conditions for the IGL problem and correspondingly propose sample-based and cluster-based strategies, which generate a graph within a restricted size for updating the model to achieve efficient learning at each time. Experimental results of node classification and link prediction tasks on growing graphs show that the proposed methods achieve satisfying performance with high efficiency. Future works may further step into our modulized graph learning process and adapt IGL to growing graphs with real-world timestamps.

## REFERENCES


Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with A-GEM. In *International Conference on Learning Representations*, 2019a.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 257–266, 2019.

Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *Proceedings of the 35th International Conference on Machine Learning, PMLR*, volume 80, 2018.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Frank Harary and Gopal Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–87, 1997.

Pili Hu and Wing Cheong Lau. A survey and taxonomy of graph sampling. *arXiv preprint arXiv:1308.5865*, 2013.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

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

Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 631–636, 2006.

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

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pp. 6467–6476, 2017.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

Franco Manessi, Alessandro Rozza, and Mario Manzo. Dynamic graph convolutional networks. *Pattern Recognition*, 97:107000, 2020.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B Schardl, and Charles E Leiserson. EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *AAAI*, pp. 5363–5370, 2020.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 374–382, 2019.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. *arXiv preprint arXiv:2002.07962*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2019.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Junzhou Zhao, Pinghui Wang, John CS Lui, Don Towsley, and Xiaohong Guan. Sampling online social networks by random walk with indirect jumps. *Data Mining and Knowledge Discovery*, 33 (1):24–57, 2019.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

# A More Details of Node Classification

## A.1 Configurations

We set the walking probability $p = 0.85$ in the random jump method. For the backbone model GraphSAGE, we set the minibatch size as 1000 and the neighborhood sample sizes as 25, 10, which are the same as the original paper. During the training process, we use the Adam optimizer with learning rate of 0.01 and weight decay of 0.0005. At each time, we run 100 epochs for training and validation and load the model with best validation accuracy for test. We run 50 times on Cora, Citerseer and Pubmed to report the average results with standard derivations, and 5 times on Flickr and Reddit. The experiments are conducted on one GPU of GeForce GTX 1080Ti.

### A.2 PSEUDO-CODE

Given the specified memory constraint $V_{max} = M \times C$ (number of categories), we apply the sampling and clustering inside each category with the constraint $M$. Since the labeling information is required, the above process is conducted on the training and validation set respectively. The pseudocode in Algorithm 2 shows details of generating the learned graph in node classification.

**Algorithm 2** Process of generating learned graph in node classification.

**Input:** $\mathcal{V}_t^{new}, \mathcal{E}_t^{intra}, \mathcal{G}_t(\mathcal{V}_t, \mathcal{E}_t), V_{max}, E_{max}$
**Input:** Category information $catecory(\cdot)$ for the label vertices
**Output:** The learned graph $\mathcal{G}_t^L(\mathcal{V}_t^L = \mathcal{V}_t^{L-train} \cup \mathcal{V}_t^{L-val} \cup \mathcal{V}_t^{L-test}, \mathcal{E}_t^L)$
  $\mathcal{V}_t^{new} = \mathcal{V}_t^{new-train} \cup \mathcal{V}_t^{new-val} \cup \mathcal{V}_t^{new-test}$ // get split of new vertices
  $\mathcal{V}_{t-1} = \mathcal{V}_t - \mathcal{V}_t^{new} = \mathcal{V}_{t-1}^{train} \cup \mathcal{V}_{t-1}^{val} \cup \mathcal{V}_{t-1}^{test}$ // get split of old vertices
  $\mathcal{V}_t^{L-train} \leftarrow \mathcal{V}_t^{new-train}, \mathcal{V}_t^{L-val} \leftarrow \mathcal{V}_t^{new-val}, \mathcal{V}_t^{L-test} \leftarrow \mathcal{V}_t^{new-test}$
  $\mathcal{E}_t^L \leftarrow \mathcal{E}_t^{intra}$
  **for** c = 1, 2, ..., C **do**
    $\mathcal{V}_{t-1}^c \leftarrow \{\mathbf{v} \in \mathcal{V}_{t-1} \mid category(\mathbf{v}) = c\}$
    Generate $\mathcal{V}_{t-1}^{c-train}, \mathcal{V}_{t-1}^{c-val}$ from $\mathcal{V}_{t-1}^c$ with specified method under constraints of $\frac{V_{max}}{C}, \frac{E_{max}}{C}$
    $\mathcal{V}_t^{L-train} \leftarrow \mathcal{V}_t^{L-train} \cup \mathcal{V}_{t-1}^{c-train}, \mathcal{V}_t^{L-val} \leftarrow \mathcal{V}_t^{L-val} \cup \mathcal{V}_{t-1}^{c-val}$
  **end for**
  $\mathcal{V}_t^L \leftarrow \mathcal{V}_t^{L-train} \cup \mathcal{V}_t^{L-val} \cup \mathcal{V}_t^{L-test}$
  $\mathcal{E}_t^L \leftarrow \mathcal{E}_t^L \cup \{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t \mid \mathbf{u} \in \mathcal{V}_t^L - \mathcal{V}_t^{new}, \mathbf{v} \in \mathcal{V}_t^{new}\}$
  **return** $\mathcal{G}_t^{learn}(\mathcal{V}_t^{learn}, \mathcal{E}_t^{learn})$

### A.3 BACKBONE NETWORK ARCHITECTURES

We list the architectures and parameters of the backbone networks in main experiment. GCN_Conv($f_{in}$, $f_{out}$) denotes the convolution defined in GCN with the input and output channels as $f_{in}$ and $f_{out}$, and SAGE_Conv($f_{in}$, $f_{out}$) denotes the same in GraphSAGE.

| | Cora | Citeseer | Pubmed | Flickr | Reddit |
|---|---|---|---|---|---|
| Layer1 | Dropout(p=0.5)<br>GCN_Conv(1433, 16)<br>ReLU() | Dropout(p=0.5)<br>GCN_Conv(3707, 64)<br>ReLU() | SAGE_Conv(500, 64)<br>ReLU() | SAGE_Conv(500, 256)<br>ReLU() | SAGE_Conv(602, 128)<br>ReLU() |
| Layer2 | Dropout(p=0.5)<br>GCN_Conv(16, 7)<br>SoftMax() | Dropout(p=0.5)<br>GCN_Conv(64, 6)<br>SoftMax() | Dropout(p=0.5)<br>SAGE_Conv(64, 3)<br>SoftMax() | Dropout(p=0.5)<br>SAGE_Conv(256, 2)<br>SoftMax() | Dropout(p=0.5)<br>SAGE_Conv(128, 41)<br>SoftMax() |

The architectures of backbones for Pubmed in ablation study are as follows.

| | GCN | GAT | GIN |
|---|---|---|---|
| Layer1 | Dropout(p=0.5)<br>GCN_Conv(1433, 64)<br>ReLU() | Dropout(p=0.6)<br>GAT_Conv(1433, 64)<br>ELU() | Dropout(p=0.5)<br>GIN_Conv(1433, 64)<br>ReLU() |
| Layer2 | Dropout(p=0.5)<br>GCN_Conv(64, 3)<br>SoftMax() | Dropout(p=0.6)<br>GAT_Conv(64, 3)<br>SoftMax() | Dropout(p=0.5)<br>GIN_Conv(64, 3)<br>SoftMax() |

### A.4 CLASSIFICATION ACCURACY THROUGHOUT THE TIME

We show the classification accuracy on the entire observed graph at each time. It is observed that in the starting time of growing graph with larger $T$ and the whole time of growing graph with smaller $T$, the classification accuracy shows uptrend of all methods. This is because in such stage, the training samples are not enough for models to capture patterns of each categories. While in the later time, results of finetuning show downtrend or unstable performance due to the forgetting of learned patterns. In contrast, results of joint training could maintain stable performance for it repeats to train on all the previous data. For our proposed methods, sample-based strategy solves the problem of

forgetting in the later period of time, while leave larger distances to joint training, compared with cluster-based strategy.

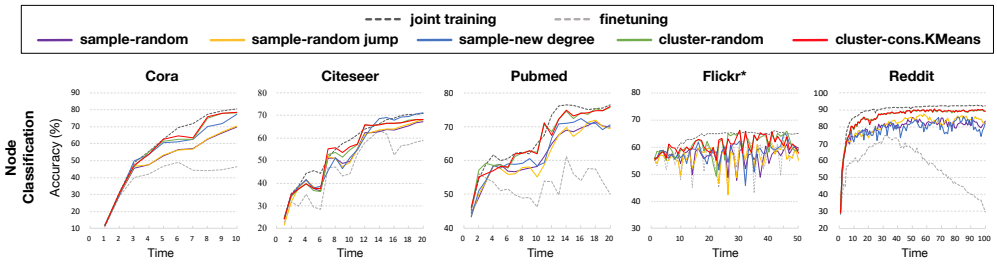


Figure 5: Classification accuracy on the entire observed graph at each time.

# B More Details of Link Prediction

## B.1 Configurations

For training with the reconstruction loss on large graphs, we sample 1,000 positive and negative edges respectively for computing the loss in each iteration. Settings of backbone models are the same as node classification. During the training process, we use the Adam optimizer with learning rate of 0.01 and weight decay of 0.0005. At each time, we run 200 epochs for training and validation and load the model with best validation accuracy for test. We run 50 times on Cora, Citerseer and Pubmed datasets to get the average results, and 5 times on Flickr and Reddit datasets. The experiments are conducted on one GPU of GeForce GTX 1080Ti.

## B.2 Negative Edge Sampling

The reconstruction loss requires sampled negative edges. For training on small graphs, the negative edges are directly sampled from those unconnected vertex pairs in each iteration. However, such sampling process is time-consuming on large graphs, thus we generate a set of negative edges in advance and randomly sample required number of edges from it. The algorithm for generating negative edge set is in Algorithm 3.

**Algorithm 3** Negative edge sampling on large graphs.

**Input:** the positive edges $\mathcal{E}^{pos}$ and corresponding vertices $\mathcal{V}$
  initialize $\mathcal{E}^{neg} \leftarrow \phi$
  **for** $(\mathbf{v}_1, \mathbf{v}_2)$ in $\mathcal{E}^{pos}$ **do**
    randomly select $\mathbf{v}_2'$ from $\mathcal{V}$
    $\mathcal{E}^{neg} \leftarrow \mathcal{E}^{neg} \cup \{(\mathbf{v}_1, \mathbf{v}_2')\}$
  **end for**
  **return** $\mathcal{E}^{neg}$

It is noted that we do not check if the sampled edge belongs to the positive set. Because the checking process is also time-consuming and the possibility of sampling a positive edge generally is quite low on large graphs. Since the cluster-based strategy provides new edges connecting clusters, we also apply negative sampling for those edges with above algorithm. For both small and large graphs, we generate negative cluster edges during the preprocessing and merge them with normal negative edges in training. Specially, we find that it shows better performance for the Pubmed dataset to remove the negative cluster edges in training, and we report the results in the main experiments.

## B.3 Network Architectures

We list the architectures and parameters of the backbone networks in main experiment. The backbone generates embeddings of vertices that are used to estimate the existence probability of edges.

| | Cora | Citeseer | Pubmed | Flickr | Reddit |
|---|---|---|---|---|---|
| Layer1 | GCN_Conv(1433, 64)<br>ReLU() | GCN_Conv(3707, 128)<br>ReLU() | SAGE_Conv(500, 128)<br>ReLU() | SAGE_Conv(500, 512)<br>ReLU() | SAGE_Conv(602, 256)<br>ReLU() |
| Layer2 | GCN_Conv(128, 64) | GCN_Conv(128, 64) | Dropout(p=0.5)<br>SAGE_Conv(128, 64) | Dropout(p=0.5)<br>SAGE_Conv(512, 256) | Dropout(p=0.5)<br>SAGE_Conv(128, 64) |

### B.4 Prediction AUC throughout the time

We show the AUC on the entire observed graph at each time, similar to the node classification. However, different trends are shown in the results. All the methods could reach a relatively high performance compared to their average results, since the task of link prediction might be easier in smaller subgraphs than the node classification. On the Pubmed where cluster-based strategy outperforms joint training, we observe that it reaches higher in the later period of time, when the generated cluster-graph could include more edges that are not in the original graph.

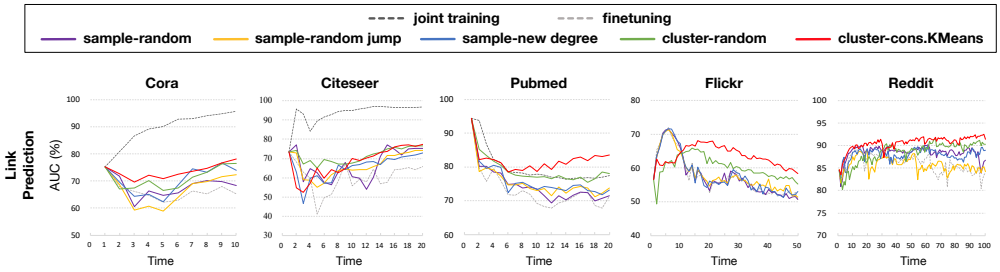


Figure 6: AUC on the entire observed graph at each time.

## C Datasets

### C.1 Statistics

Table 4 shows statistics of the benchmark datasets used in the experiments. Besides the basic information including graph size, feature dimensions and labels, we list the split of training, validation and test sets. For the Cora, Citeseer and Pubmed, we follow the standard fixed split in GCN (Kipf & Welling, 2017). For the Flickr and Reddit, to follow the similar split fashion, we uniformly sample 200 and 100 vertices in each category for training, and sample fixed number of vertices as the validation and test sets. Specifically, we generate a two-category version of Flickr, noted as "Flickr*", in experiments of node classification. Since most vertices in Flickr are in label of category 4 or 6, when split into groups, the evaluation metric turns to be better on models overfitting to these two categories, which influences the comparison of methods. Thus, we remove vertices belong to other categories and related edges, and use the rested graph as "two-category version" of Flickr. For the link prediction, we use the original Flickr dataset without category information. All the dataset are collected by the library of PyTorch Geometric (`https://github.com/rusty1s/pytorch_geometric`).

### C.2 Changes of Degrees in Growing Graph

To know details of graph changes during the growth, we further show the average degree of the subgraphs and entire graph in Figure 7. For the subgraphs, since it is randomly cut off from the graph, the average degree fluctuates throughout all tasks within in low range, which brings difficulty for learning directly on the subgraphs. The average degree of the entire graph continually grows, which is attributed to the appearance of inter-subgraph edges. And missing of such inter-edges influence on the graph learning results.

Table 4: Dataset statistics and basic settings.

| | Cora | Citeseer | Pubmed | Flickr* | Flickr | Reddit |
|---|---|---|---|---|---|---|
| **# Nodes** | 2,708 | 3,327 | 19,717 | 60,736 | 89,250 | 232965 |
| **# Edges** | 5,429 | 4,732 | 44,338 | 409,564 | 899,756 | 11,606,919 |
| **# Features** | 1,433 | 3,707 | 500 | 500 | 500 | 602 |
| **# Classes** | 7 | 6 | 3 | 2 | 7 | 41 |
| **# Training Nodes** | 140 | 120 | 60 | 400 | - | 4,100 |
| **# Validation Nodes** | 500 | 500 | 500 | 2,000 | - | 12,300 |
| **# Test Nodes** | 1,000 | 1,000 | 1,000 | 30,000 | - | 50,000 |

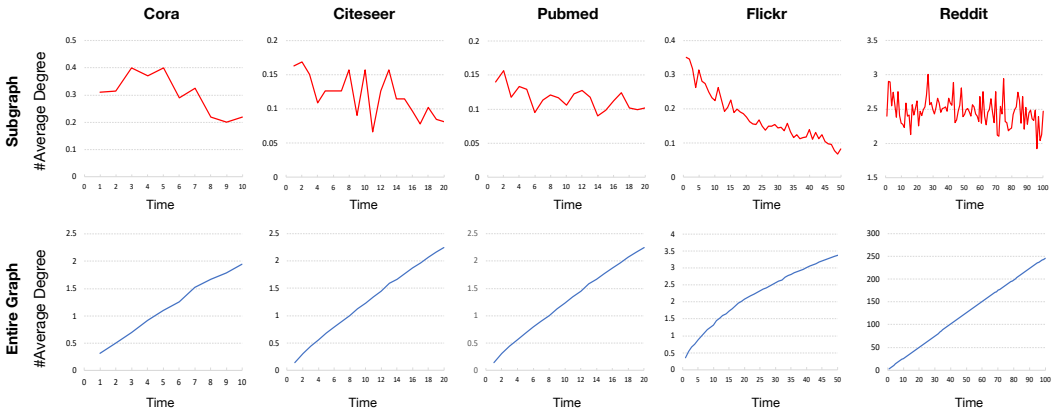


Figure 7: The average degree of subgraph and entire graph throughout tasks in benchmarks.

## D Details of Compared Methods

Since existing incremental learning methods are designed for incremental categories or domains, we adapt representative methods to our IGL scenario for comparison. In this section, we present brief introductions to these methods and describe how we adapt them for incremental graph learning.

**LwF** (Li & Hoiem, 2017) presents a distillation loss that forces the updated model to generate similar outputs with the previous model using KL-Divergence. Before the trainiang at each time, it copies the parameters of model learned on previous data, and uses the model as a "teacher-model" to restrict the update of current model.

**EWC** (Kirkpatrick et al., 2017) presents an regularization loss term to penalize the update of parameters based on their importance to previous tasks. The importance of parameters are estimated by the Fisher information matrix.

**iCaRL** (Rebuffi et al., 2017) firstly addresses the problem of incremental class learning on neural networks. It is a rehearsal-based method that store previous samples, named as exemplars, with restricted memory constraints. We adapt its way to selecting exemplars into our sample-based strategy, which calculate the mean value of each category and select samples closer to the centers. Such strategy of sampling exemplars is also accepted in the following methods, *e.g.* EEIL (Castro et al., 2018) and BiC (Wu et al., 2019).

**TEM** (Chaudhry et al., 2019b) explores the different ways of updating the exemplars set within specified memory, including reservoir, ringbuffer, k-Means and MoF. Since k-Means and MoF are based on similar ideas to the iCaRL, we adapt the ringbuffer strategy that follows FIFO rule to preserve the last several samples of each category.

**A-GEM** follows and further improves GEM (Lopez-Paz & Ranzato, 2017) to restrict optimization of parameters based on the direction of gradients, where a reference gradient is computed by a random sampled subset of previous samples.

Among the compared methods, LwF and EWC require hyper-parameters to control the weight of regularization loss terms, which are determined by grid search in IGL settings.

# E PROOFS

## E.1 PROOF TO THEOREM 4.1

Let $|\mathcal{V}_t| = n, |\mathcal{N}_t(\mathbf{v})| = m, |\mathcal{V}_t^L| = k$, we denote $L$ as the number of vertices in $\mathcal{N}_t(\mathbf{v})$ that are included in $\mathcal{V}_t^L$, *i.e.* $L = |\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|$ and $0 \leq L \leq \min(m, k)$. Then

$$\begin{aligned}
&\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L, L = l\right)\right) \\
&= \Sigma_{l=0}^{\min(m,k)} P(L = l) * \mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L, L = l\right),
\end{aligned} \tag{7}$$

Let $S(\mathcal{N}_t(\mathbf{v}), l)$ includes all the possible combination sets with length of $l$ from $\mathcal{N}_t(\mathbf{v})$, *i.e.* $S(\mathcal{N}_t(\mathbf{v}), l) = \{\mathcal{S} \subseteq \mathcal{N}_t(\mathbf{v}) \wedge |\mathcal{S}| = l\}$. Since $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_t, P(\mathbf{u}_1 \in \mathcal{V}_t^L) = P(\mathbf{u}_2 \in \mathcal{V}_t^L)$, we get

$$\begin{aligned}
&\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L, L = l\right) \\
&= \Sigma_{\mathcal{S} \in S(\mathcal{N}_t(\mathbf{v}), l)} \frac{1}{C(m, l)} * \frac{\Sigma_{\mathbf{u} \in \mathcal{S}} \mathbf{u}}{l} \\
&= \frac{1}{l * C(m, l)} \Sigma_{\mathbf{u} \in \mathcal{N}_t(\mathbf{v})} C(m-1, l-1)\mathbf{u} \\
&= \frac{\Sigma_{\mathbf{u} \in \mathcal{N}_t(\mathbf{v})} \mathbf{u}}{m} = agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right)
\end{aligned} \tag{8}$$

where $C(\cdot, \cdot)$ denotes the combination. Substitute (8) for (7), we get

$$\begin{aligned}
&\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \mid \mathbf{v} \in \mathcal{V}_t^L\right) \\
&= \Sigma_{l=0}^{\min(m,k)} P(L = l) * agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right) \\
&= agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\right)
\end{aligned} \tag{9}$$

## E.2 PROOF TO THEOREM 4.2

We follow the basic notations in the above proof. When $\mathbf{v} \in \mathcal{V}_t^L$

$$\begin{aligned}
&agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right) \cap \mathcal{V}_t^L\right) \\
&= \frac{1}{L} \Sigma_{\mathbf{u} \in \mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L} \mathbf{u} \\
&= \frac{1}{L} \Sigma_{\mathbf{u} \in \mathcal{N}_{t-1}(\mathbf{v}) \cap \mathcal{V}_t^L} \mathbf{u} + \frac{1}{L} \Sigma_{\mathbf{u} \in (\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v})) \cap \mathcal{V}_t^L} \mathbf{u}
\end{aligned} \tag{10}$$

Since $\forall \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_{t-1}, P(\mathbf{u}_1 \in \mathcal{V}_t^L) = P(\mathbf{u}_2 \in \mathcal{V}_t^L)$, similar to Theorem 4.1, we let $|\mathcal{N}_{t-1}| = m'$ and $l$ be the number of sampled vertices from old data, then

$$
\begin{aligned}
&\mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in\mathcal{N}_{t-1}(\mathbf{v})\cap\mathcal{V}_t^L}\mathbf{u} \mid \mathbf{v}\in\mathcal{V}_t^L\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in\mathcal{N}_{t-1}(\mathbf{v})\cap\mathcal{V}_t^L}\mathbf{u} \mid \mathbf{v}\in\mathcal{V}_t^L, L = l + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v})|\right)\right) \\
&= \Sigma_{l=0}^{\min(m',k)} P(L = l + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}))|) * \left(\Sigma_{\mathcal{S}\in S(\mathcal{N}_{t-1}(\mathbf{v}),l)}\frac{1}{C(m',l)} * \frac{\Sigma_{\mathbf{u}\in\mathcal{S}}\mathbf{u}}{L}\right) \\
&= \Sigma_{l=0}^{\min(m',k)} P(L = l + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}))|) * \left(\frac{1}{L * C(m',l)}\Sigma_{\mathbf{u}\in\mathcal{N}_{t-1}(\mathbf{v})}C(m'-1,l-1)\mathbf{u}\right) \\
&= \Sigma_{l=0}^{\min(m',k)} P(L = l + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}))|) * \frac{l}{L} * \frac{\Sigma_{\mathbf{u}\in\mathcal{N}_{t-1}(\mathbf{v})}\mathbf{u}}{m'} \\
&= \mathbb{E}(\frac{l}{l + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}))|})agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v})) \\
&= \mathbb{E}(\frac{|\mathcal{N}_{t-1}(\mathbf{v})\cap\mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|})agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v}))
\end{aligned}
\tag{11}
$$

Since $\forall \mathbf{u} \in \mathcal{V}_t^{new}, P(\mathbf{u} \in \mathcal{V}_t^L) = 1$, we get $(\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v})) \cap \mathcal{V}_t^L = \mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v})$, then

$$
\begin{aligned}
&\mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in(\mathcal{N}_t(\mathbf{v})-\mathcal{N}_{t-1}(\mathbf{v}))\cap\mathcal{V}_t^L}\mathbf{u}\right) \\
&= \mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in\mathcal{N}_t(\mathbf{v})-\mathcal{N}_{t-1}(\mathbf{v})}\mathbf{u}\right) \\
&= \mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in\mathcal{N}_t(\mathbf{v})}\mathbf{u}\right) - \mathbb{E}\left(\frac{1}{L}\Sigma_{\mathbf{u}\in\mathcal{N}_{t-1}(\mathbf{v})}\mathbf{u}\right) \\
&= \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)agg(\mathbf{v}, \mathcal{N}_t(\mathbf{v})) - \mathbb{E}\left(\frac{|\mathcal{N}_{t-1}(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v}))
\end{aligned}
\tag{12}
$$

Substitute (11) and (12) for (10), we get

$$
\begin{aligned}
&\mathbb{E}\left(agg\left(\mathbf{v}, \mathcal{N}_t\left(\mathbf{v}\right)\cap\mathcal{V}_t^L\right)\right) \\
&= \mathbb{E}(\frac{|\mathcal{N}_{t-1}(\mathbf{v})\cap\mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|})agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v})) + \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)agg(\mathbf{v}, \mathcal{N}_t(\mathbf{v})) \\
&- \mathbb{E}\left(\frac{|\mathcal{N}_{t-1}(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v})) \\
&= \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)agg(\mathbf{v}, \mathcal{N}_t(\mathbf{v})) - \mathbb{E}(\frac{|\mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|})agg(\mathbf{v}, \mathcal{N}_{t-1}(\mathbf{v}))
\end{aligned}
\tag{13}
$$

Theorem 4.2 proved.

We then prove that $\lambda_1 + \lambda_2 = 1$, where $\lambda_1 = \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right), \lambda_2 = -\mathbb{E}\left(\frac{|\mathcal{N}_{t-1}(\mathbf{v})-\mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right)$.

$$
\begin{aligned}
\lambda_1 + \lambda_2 &= \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})|}{|\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}\right) - \mathbb{E}\left(\frac{|\mathcal{N}_{t-1}\left(\mathbf{v}\right) - \mathcal{V}_t^L|}{|\mathcal{N}_t\left(\mathbf{v}\right)\cap\mathcal{V}_t^L|}\right) \\
&= \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v})| - |\mathcal{N}_{t-1}\left(\mathbf{v}\right) - \mathcal{V}_t^L|}{|\mathcal{N}_t\left(\mathbf{v}\right)\cap\mathcal{V}_t^L|}\right) \\
&= \mathbb{E}\left(\frac{|\mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L| + |\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L| + |\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L| - |\mathcal{N}_{t-1}\left(\mathbf{v}\right) - \mathcal{V}_t^L|}{|\mathcal{N}_t\left(\mathbf{v}\right)\cap\mathcal{V}_t^L|}\right) \\
&= \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L| + |\mathcal{N}_t(\mathbf{v})\cap\mathcal{V}_t^L|}{|\mathcal{N}_t\left(\mathbf{v}\right)\cap\mathcal{V}_t^L|}\right)
\end{aligned}
\tag{14}
$$

Since $\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}) \subseteq \mathcal{V}_t^L$, we get $|\mathcal{N}_t(\mathbf{v}) - \mathcal{N}_{t-1}(\mathbf{v}) - \mathcal{V}_t^L| = 0$, then

$$\lambda_1 + \lambda_2 = \mathbb{E}\left(\frac{|\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|}{|\mathcal{N}_t(\mathbf{v}) \cap \mathcal{V}_t^L|}\right) = 1 \tag{15}$$

### E.3 Proof to Theorem 4.3

(i) When $\forall \mathbf{u} \in \mathcal{V}_{t-1}, P(\mathbf{u} \in \Delta\mathcal{V}_t^L) = 1$, we get $\Delta\mathcal{V}_t^L = \mathcal{V}_{t-1}$ and $\mathcal{V}_t^L = \mathcal{V}_t$. Then obviously

- $\forall \mathbf{u}, \mathbf{v} \in \mathcal{V}_{t-1}, P(\mathbf{u} \in \mathcal{V}_t^L) = P(\mathbf{v} \in \mathcal{V}_t^L) = 1$.
- $\mathcal{E}_t^L = \mathcal{E}_t, |\Delta\mathcal{E}_t^L \cap \mathcal{E}_t^{inter}| = |\mathcal{E}_t^{inter}| = \max_{\Delta\mathcal{E}_t^L} |\Delta\mathcal{E}_t^L \cap \mathcal{E}_t^{inter}|$.

(ii) When the unbiased condition and edge-preserved condition hold, we assume that the maximum target is restricted by the specified size $V_{max}$ and $E_{max}$. Thus, we consider $P(\mathbf{u} \in \mathcal{V}_t^L) \propto |\{(\mathbf{u}, \mathbf{v}) \in \mathcal{E}_t^{inter} \mid \mathbf{v} \in \mathcal{V}_t^{new}\}| = D_t(\mathbf{u}) - D_{t-1}(\mathbf{u})$. Since $\exists \mathbf{u}_1, \mathbf{u}_2 \in \mathcal{V}_{t-1}$, *s.t.* $D_t(\mathbf{u}_1) - D_{t-1}(\mathbf{u}_1) \neq D_t(\mathbf{u}_2) - D_{t-1}(\mathbf{u}_2)$, we get $P(\mathbf{u}_1 \in \mathcal{V}_t^L) \neq P(\mathbf{u}_2 \in \mathcal{V}_t^L)$, which conflicts with the unbiased condition. So the assumption does not hold.

### E.4 Proof to Theorem 4.4

Theorem 4.4 represents sampling as a single observation equation $f(\cdot)$, and clustering as a group observation equation $g(\cdot, \cdot)$. Thus, the theorem suggests that cluster-based strategy provides more information quantity than sample-based strategy, and a balanced clustering works better in cluster-based strategy.

To prove the first inequality in Theorem 1, we firstly prove the following lemma that knowing the average value of the whole set provides more information than knowing value of some element.

**Lemma 1.** *Given $n$ elements $x_1, x_2, ..., x_n$ $(n \geq 1)$ with possible value 0 or 1. Follow the definition of $f(\cdot)$ and $g(\cdot, \cdot)$, $\forall i \in [1, n]$,*

$$S(g(1, n)) \leq S(f(i)). \tag{16}$$

*Proof.* We calculate the value of entropy based on the definition that $S = k_B \log \Omega$, where $\Omega$ is the number of mircostates. We omit the constant $k_B$ for simplicity. Then for the left of the inequality knowing definite value of one element, $\Omega = 2^{n-1}$ and $S(f(i)) = \log 2^{n-1}$. For the right of the inequality, suppose $g(1, n) : \Sigma_{j=1}^n x_j = v$, then $\Omega = C(n, v) = \frac{n!}{n!(n-v)!}$ where $C(\cdot, \cdot)$ denotes the combination. It is known that $C(n, v) \leq C(n, \lfloor \frac{n}{2} \rfloor)$. Using the inequality conducted from the *Stirling's approximation* that

$$\sqrt{2\pi} n^{n+\frac{1}{2}} e^{-n} \leq n! \leq e\, n^{n+\frac{1}{2}} e^{-n}, \tag{17}$$

we get

$$\begin{aligned} C(n, v) \leq C(n, \lfloor n/2 \rfloor) &= \frac{n!}{\lfloor n/2 \rfloor! \lceil n/2 \rceil!} \\ &\leq \frac{e\, n^{n+\frac{1}{2}} e^{-n}}{(\sqrt{2\pi} \lfloor n/2 \rfloor^{\lfloor n/2 \rfloor + \frac{1}{2}} e^{-\lfloor n/2 \rfloor})(\sqrt{2\pi} \lceil n/2 \rceil^{\lceil n/2 \rceil + \frac{1}{2}} e^{-\lceil n/2 \rceil})} \\ &\leq \frac{e\, n^{n+\frac{1}{2}}}{2\pi (\frac{n}{2})^{n+1}} = \frac{e\, 2^{n+1}}{2\pi\sqrt{n}} = \frac{2e}{\pi\sqrt{n}} 2^{n-1} \end{aligned} \tag{18}$$

For $n \geq 3$, $\frac{2e}{\pi\sqrt{n}} < 1$, then $C(n, v) < 2^{n-1}$. When $n = 1, 2$, it is easy to know from calculation that $C(n, v) \leq 2^{n-1}$. Thus $\forall n \geq 1$,

$$S(g(1, n)) = \log C(n, v) \leq \log 2^{n-1} = S(f(i)) \tag{19}$$

Now we prove the first inequality in Theorem 1 that

$$S(g(1, \frac{n}{r}), g(\frac{n}{r} + 1, \frac{2n}{r}), ..., g(\frac{(r-1)n}{r} + 1, n)) \leq S(f(1), f(2), ..., f(r)) \tag{20}$$

Split the $n$ elements with length of $\frac{n}{r}$ and denote the number of microstates of each group as $\Omega_i(e_i), i = 0, 1, ..., (r-1)$, where $e_i$ is the equation within this group. Then based on Lemma 1, we get

$$\Omega_i(g(\frac{in}{r}+1, \frac{(i+1)n}{r})) \leq 2^{\frac{n}{r}-1}, i = 0, 1, ..., (r-1) \tag{21}$$

Then we sum the above inequalities from 0 to $(r-1)$ as

$$\begin{aligned} &S(g(1, \frac{n}{r}), g(\frac{n}{r}+1, \frac{2n}{r}), ..., g(\frac{(r-1)n}{r}+1, n)) \\ &= \log \prod_{i=0}^{r-1} \Omega_i \leq \log (2^{\frac{n}{r}-1})^r \\ &= \log 2^{n-r} = S(f(1), f(2), ..., f(r)) \end{aligned} \tag{22}$$

The above proof shows that the cluster-based strategy offering average value provides more information that the sample-based strategy offering single element value, which is stated by the first inequality in Theorem 1.

We restate the second inequality in Theorem 1:

$$\begin{aligned} &\mathbb{E}(S(g(1, \frac{n}{r}), g(\frac{n}{r}+1, \frac{2n}{r}), ..., g(\frac{(r-1)n}{r}+1, n))) \leq \\ &\mathbb{E}(S(g(1, L_1), g(L_1+1, L_2), ..., g(L_{r-1}+1, n))), \end{aligned} \tag{23}$$

where $\mathbb{E}()$ is the expectation of entropy. Given $n$ elements $x_1, ..., x_n$, the possible value $s = \Sigma_{i=1}^n x_i$ are from 0 to n, with $C(n, s)$ conditions out of the total $2^n$ conditions. Then the expectation of $\Omega$ knowing the average value is

$$\mathbb{E}(\Omega(g(1, n)) = \Sigma_{s=0}^n \frac{C(n,s)}{2^n} C(n, s) = 2^{-n} \Sigma_{s=0}^n C(n, s)^2 \tag{24}$$

We define the discrete function $p(n) = 2^{-n} \Sigma_{s=0}^n C(n, s)^2$. By computing the expectation of $\Omega_i, i = 1, ..., r$ from $r$ groups in (23), we get

$$\begin{aligned} &\mathbb{E}(S(g(1, \frac{n}{r}), g(\frac{n}{r}+1, \frac{2n}{r}), ..., g(\frac{(r-1)n}{r}+1, n))) \\ &= \log \prod_{i=0}^{r-1} \Omega_i(g(\frac{in}{r}+1, \frac{(i+1)n}{r})) \\ &= \log \prod_{i=0}^{r-1} p(\frac{n}{r}) = r \log p(\frac{n}{r}), \end{aligned} \tag{25}$$

and similarly

$$\mathbb{E}(S(g(1, L_1), g(L_1+1, L_2), ..., g(L_{r-1}+1, n))) = \Sigma_{i=1}^r \log p(l_i) \tag{26}$$

Then if the function $\log p(n)$ is a convex function, using the convexity inequality $\forall \lambda_i > 0, \Sigma \lambda_i = 1, \log p(\Sigma_{i=1}^r \lambda_i x_i) \leq \Sigma_{i=1}^r \lambda_i \log p(x_i)$, and set $\lambda_i = \frac{1}{r}, x_i = l_i$, we could get

$$\log p(\frac{n}{r}) = \log p(\Sigma_{i=1}^r \frac{l_i}{r}) \leq \Sigma_{i=1}^r \frac{\log p(l_i)}{r} \tag{27}$$

Then $r \log p(\frac{n}{r}) \leq \Sigma_{i=1}^r \log p(l_i)$ and the inequality (23) holds. By calculation, $q(n) = \log p(n) = \log \frac{2^n (n-\frac{1}{2})!}{\sqrt{\pi} n!}$ and the 2nd order difference $\Delta q(n) \geq 0$, thus $\log p(n)$ is convex and the proved inequality holds.

## F MORE VISUALIZATIONS

Figure 8 shows the visualization results of all methods in 4.4. We further visualize the node classification results at each time in Figure 9, showing the correctly and wrongly classified nodes in green and blue respectively. Red nodes indicate the current training nodes and we do not draw the cluster nodes in our cluster-based strategy. Nodes have not appeared until now are in gray. Then two facts could be observed in each row of the visualization:

- Some newly appeared nodes are correctly classified without connecting to the training nodes, which suggests the inherited knowledge from previously trained model.
- Some blue nodes in previous task turn to be green in following tasks, which suggests that new vertices and edges may rectify the error in previous learning process.

When comparing among columns, we show the effectiveness of proposed methods together with baselines. Our methods achieve comparable performance with the joint training method with less cost of computation and memory.

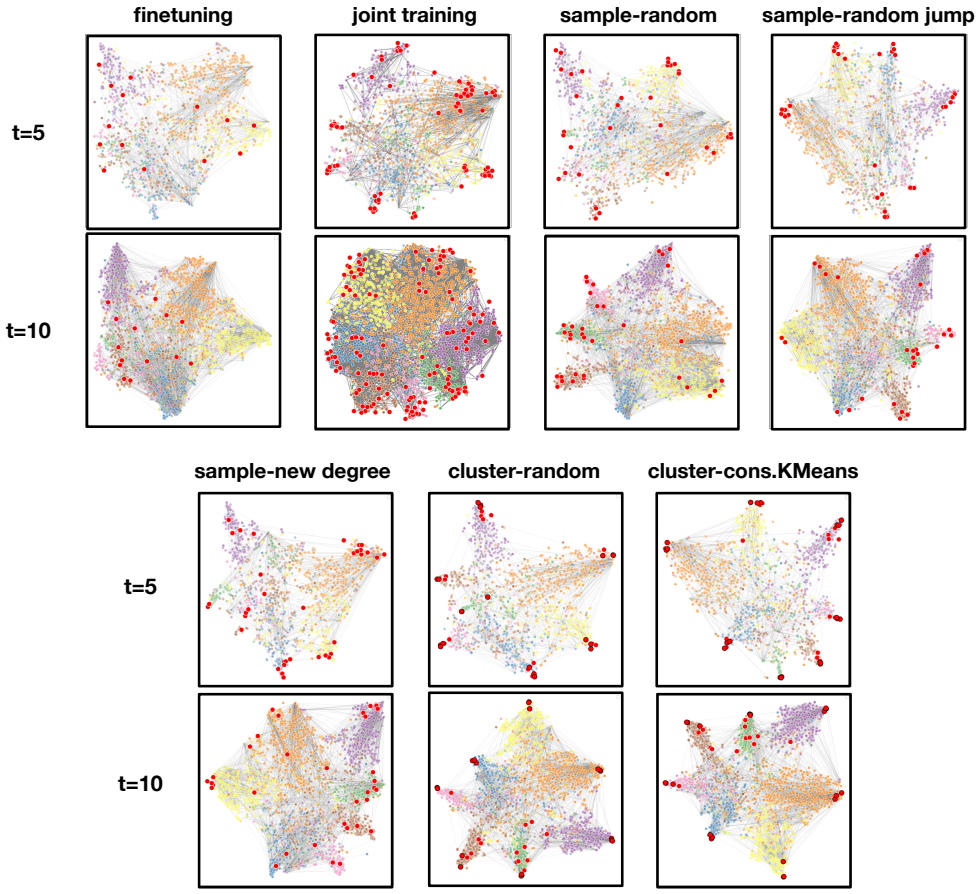


Figure 8: Feature visualization using t-SNE of all methods training node classification on Cora.

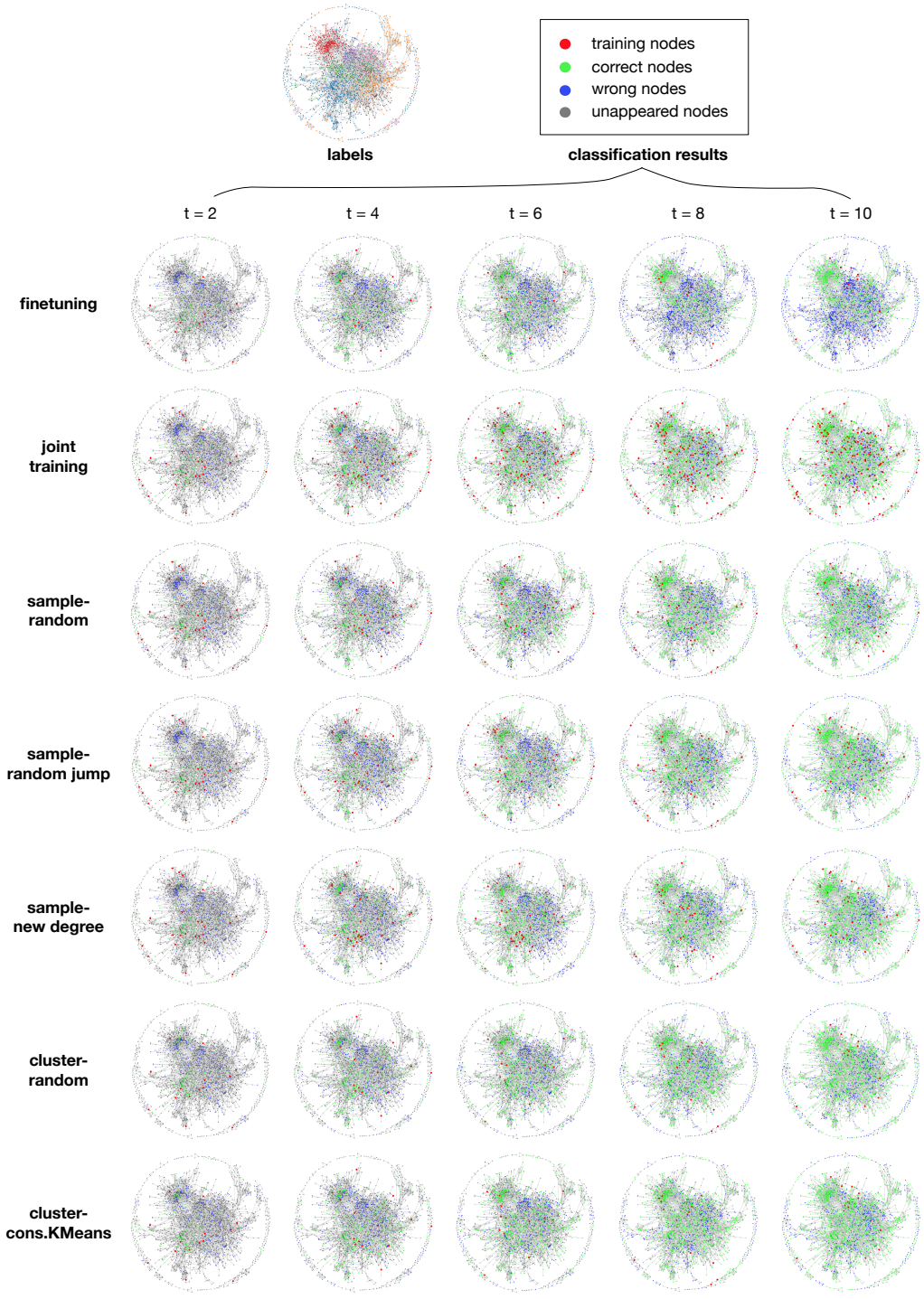


Figure 9: **Visualization of node classification results on Cora.** We show the results at time 2, 4, 6, 8, 10 in one row for each method. For the appeared nodes, we draw the training nodes in red, and draw nodes classified into correct/wrong category in green/blue. Then unappeared nodes until now are in gray.