Influential Exemplar Replay for Incremental Learning in Recommender Systems

Xinni Zhang[1], Yankai Chen[1], Chenhao Ma[2], Yixiang Fang[2], Irwin King[1]

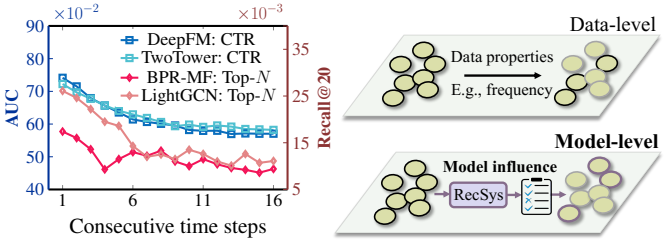
[1]The Chinese University of Hong Kong
[2]The Chinese University of Hong Kong, Shenzhen
{xnzhang23, ykchen, king}@cse.cuhk.edu.hk, {machenhao, fangyixiang}@cuhk.edu.cn

Abstract

Personalized recommender systems have found widespread applications for effective information filtering. Conventional models engage in knowledge mining within the *static* setting to reconstruct singular historical data. Nonetheless, the *dynamics* of real-world environments are in a constant state of flux, rendering acquired model knowledge inadequate for accommodating emergent trends and thus leading to notable recommendation performance decline. Given the typically prohibitive cost of exhaustive model retraining, it has emerged to study *incremental learning for recommender systems* with ever-growing data. In this paper, we propose an effective model-agnostic framework, namely **INF**luential **E**xemplar **R**eplay (INFER). INFER facilitates recommender models in retaining the earlier assimilated knowledge, e.g., users' enduring preferences, while concurrently accommodating evolving trends manifested in users' new interaction behaviors. We commence with a vanilla implementation that centers on identifying the most representative data samples for effective consolidation of early knowledge. Subsequently, we propose an advanced solution, namely $\text{INFER}_{\text{ONCE}}$, to optimize the computational overhead associated with the vanilla implementation. Extensive experiments on four prototypical backbone models, two classic recommendation tasks, and four widely used benchmarks consistently demonstrate the effectiveness of our method as well as its compatibility for extending to several incremental recommender models.

Introduction

To tackle overwhelming information overload, recommender systems play a crucial role in online services, e.g., Web search, news recommendation, and E-commerce advertising. Modern algorithms exploit knowledge mining from similar user correlations, e.g., collaborative filtering (Rendle et al. 2012; He et al. 2017; Chen et al. 2023c) and social regularization (Ma et al. 2011), to content-based analysis, e.g., multimodal understanding (Zheng et al. 2018; Gupta et al. 2020) and auxiliary knowledge supplement (Wang et al. 2019a,b; Chen et al. 2022c,b; Zhang et al. 2022a). With the advancements of deep learning techniques, these recommender models exhibit strong capability in learning complex and latent features, demonstrating the performance superiority for realistic deployment (Covington, Adams, and Sargin 2016; Guo et al. 2017; Ying et al. 2018).



(a) Performance degradation (b) Replay sample retrieval

Figure 1: Illustration of INFER motivations.

Derived from conventional neural models, these recommender algorithms are mostly crafted within the *static* setting. In such learning frameworks, models acquire knowledge from singular and fixed training data and subsequently extrapolate insights to forthcoming data, with the presumption of identical distribution. However, such a "once-for-all" strategy may prove unsuitable for intricate real-world *dynamic* settings, featuring constantly emerging objects (e.g., users and items) and evolving engagements. Consequently, as shown in Figure 1a, the cessation of learning data updates impedes recommender models to assimilate new trends and knowledge (Xu et al. 2020), leading to performance degradation for proper recommendations.

To circumvent the expensive model retraining, *incremental learning* has recently exhibited promising advantages (Zenke, Poole, and Ganguli 2017; Prabhu, Torr, and Dokania 2020; Wang, Zhang, and Coates 2021; He et al. 2023a). Generally, this learning paradigm enables neural models to continually adapt to evolving data for new knowledge integration, with the primary focus to alleviate the tendency of forgetting previously learned information, a.k.a., *catastrophic forgetting* problem (McCloskey and Cohen 1989), during model incremental updating. Despite the recent progress in Computer Vision (Hung et al. 2019; Yuan et al. 2021; Bonicelli et al. 2022; Prabhu, Torr, and Dokania 2020), the exploration of incremental learning for recommender systems remains relatively nascent. The major challenges are twofold. (1) These methods are for incremental data either with different *data classes* or for different *visual tasks*, whereas the

incremental interaction data flow is for unveiling the shifts of users' preferences and hidden intentions. (2) Moreover, different from images that are essentially isolated objects, incremental interaction data encompass pairs of users and items. Incrementally learning such interactive data thereby affects the feature updating of bipartite parties via the mutual propagation of collaborative filtering signals for recommendation. Therefore, it has emerged to specifically develop incremental learning for recommender systems with a few recent attempts at graph neural methodologies (Xu et al. 2020; Wang, Zhang, and Coates 2021; Ahrabian et al. 2021).

While the aforementioned pioneer works rely on graph-based modeling, in this paper, we propose an effective model-agnostic incremental learning framework, namely **INF**luential **E**xemplar **R**eplay (INFER), that is compatible with more recommender backbones. Generally, our methodology is rooted in the *experience replay* strategy (Rebuffi et al. 2017; Zenke, Poole, and Ganguli 2017), which interleaves a small proportion of early data samples with new data for incremental model training. By doing so, the updated model gets to revisit and learn from its past experiences, preventing the loss of earlier learned knowledge, e.g., users' long-term interests or persistent preferences. Intuitively, the core of experience replay lies in finding high-quality data samples. To provide effective and compact replay samples, we propose to retrieve *influential exemplars* from historical interaction data. Different from straightforward strategies (Vitter 1985; Welling 2009; Rebuffi et al. 2017; Prabhu, Torr, and Dokania 2020; Ahrabian et al. 2021) that infer important samples based on *data-level properties*, e.g., frequencies or diversities, as shown in Figure 1b, our distinctive influential exemplars embody the elements that directly impact the *model-level performance* across various recommendation tasks. Our vanilla implementation, denoted by $\text{INFER}_{\text{Vanilla}}$, is underpinned by the influence function of Robust Statistics (Koh and Liang 2017). We further provide an advanced solution, i.e., **O**ne-step i**N**fluen**C**e **E**stimation denoted by $\text{INFER}_{\text{ONCE}}$, to optimize the efficiency and instability bottlenecks of vanilla influence functions (Basu, Pope, and Feizi 2021; Epifano et al. 2023) for large recommender models. In empirical evaluation, we implement our methodology into four renowned recommender backbones. Experiments on four real-world benchmarks and two recommendation tasks consistently showcase the performance superiority of $\text{INFER}_{\text{ONCE}}$ with desirable generalization and efficiency. Our principal contributions are summarized as follows:

- To the best of our knowledge, we are the first to incorporate analytical influence methodology for constructing incremental recommender models, via retrieving replay exemplars directly influential to downstream tasks.
- To deal with the efficiency and instability issues of our vanilla implementation, we further propose a refined approach namely $\text{INFER}_{\text{ONCE}}$ with computation approximation and acceleration.
- Extensive experiments demonstrate the high adaptability of $\text{INFER}_{\text{ONCE}}$ across several backbone architectures. It consistently outperforms recent end-to-end incremental recommender solutions on four public benchmarks.

Related Work

Collaborative Filtering

Collaborative filtering (CF), as one classic paradigm of recommender algorithms, parameterizes users and items as embeddings to reconstruct historical interactions (Ying et al. 2018; Chen et al. 2022a; Zhang et al. 2023b; Chen et al. 2023b). Early CF models such as matrix factorization (Koren, Bell, and Volinsky 2009; Chen et al. 2020) usually decompose the user-item interaction matrix into low-dimensional embeddings. More recent neural models like DeepFM (Guo et al. 2017), TwoTower (Huang et al. 2013) and NCF (He et al. 2017) further leverage neural networks to improve modeling capability. Since user-item interactions can be naturally represented by a bipartite graph (Chen et al. 2023a), Graph Neural Networks (Hamilton, Ying, and Leskovec 2017; Velickovic et al. 2017; Zhang et al. 2022b, 2023c; Song, Zhang, and King 2023; Ma et al. 2023), as the neural architectures specifically for graph data (Li et al. 2023; Zhang et al. 2023a), have gained impressive development to be a favored formulation for CF algorithms, such as NGCF (Wang et al. 2019c), LightGCN (He et al. 2020), and SimGCL (Yu et al. 2022). All these approaches work in a *static* setting, where models are designed to be trained and deployed for the entire snapshot of interaction data. Our research deviates from this assumption to focus on the *evolving* data setting that imitates the real-life environment. Specifically, our goal is to develop models with the ability to continuously learn knowledge from new interaction data as it becomes available.

Incremental Learning

Incremental learning (IL), a.k.a., continual learning, enables intelligence models with continual knowledge acquisition ability for new environment adaptation (De Lange et al. 2021; Zenke, Poole, and Ganguli 2017; Prabhu, Torr, and Dokania 2020; Rannen et al. 2017; Aljundi, Chakravarty, and Tuytelaars 2017). IL-based methods initially garner attention in Computer Vision. The main challenge is avoiding *catastrophic forgetting* (McCloskey and Cohen 1989), which is the tendency for models to substantially forget previously learned information when updated with new data. There are three major types of IL methodologies. (1) *Weight regularization methods* (Rannen et al. 2017; Kirkpatrick et al. 2017; Wang, Zhang, and Coates 2021) aim to restrict the overall amount of parameter changes by designing additional regularization of loss terms. (2) *Parameter isolation methods* (Mallya and Lazebnik 2018; Hung et al. 2019; Yuan et al. 2021) allocate model capacity with specific focuses on enabling parameter sharing and isolation while also preventing knowledge leakage during the incremental data learning. (3) *Experience replay methods* (Rebuffi et al. 2017; Prabhu, Torr, and Dokania 2020; Zenke, Poole, and Ganguli 2017; Bonicelli et al. 2022; Zhou and Cao 2021) usually sample and reuse a subset of previously encountered data in model updating to avert forgetting issues. However, most of these methods are designed for image-processing tasks, while only a few recent attempts have studied the problem for recommender systems (Xu et al. 2020; Wang, Zhang,

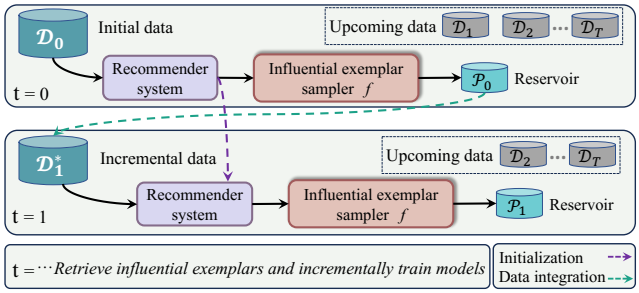


Figure 2: Training workflow of incremental recommender models with influential exemplar replay of INFER.

and Coates 2021; Ahrabian et al. 2021; He et al. 2023b; Wang et al. 2023). In this work, we focus on *experience replay* strategy and are motivated to propose a fundamental method that is compatible with various recommender backbone models.

INFER Methodology

Preliminaries

Problem Formulation. Given a user-item interaction stream list $\mathbb{D}$ with consecutive data segments: $\mathbb{D} = \{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_T\}$. At each timestamp t, the incremental data learning for recommendation accesses segment $\mathcal{D}_t$ for model updating, whilst maintaining previous knowledge from $\{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_{t-1}\}$ to prevent *catastrophic forgetting* problem.

Experience Replay Strategy. To memorize the early knowledge, *experience replay* explicitly maintains a *reservoir* to buffer a small subset of historical data. During incremental training, the buffered data are retrieved and integrated with the new data; then the underlying model is iteratively updated to mitigate its tendency to forget previously learned knowledge, e.g., users' long-term preferences.

Framework Procedure

For each timestamp t, with the original data segment $\mathcal{D}_t$, we obtain the incremental train data $\mathcal{D}_t^*$ as $\mathcal{D}_t^* := \mathcal{P}_{t-1} \cup \mathcal{D}_t$ for model updating, where $\mathcal{P}_{t-1}$ denotes the reservoir constructed at previous timestamp $t-1$. As for the construction of reservoir $\mathcal{P}_t$ at time t, K samples are filtered out from $\mathcal{D}_t^*$, which is prepared for the next incremental training. Notably, the quality of the target data reservoir directly determines the performance of incremental recommender models, which motivates us to buffer most *influential* data samples, i.e., namely *influential exemplars*. In this work, we interpret such a notion of influence as an alteration of data occurrence that perturbs recommendation performance. Specifically, our proposed framework can be abstracted by the function f as follows:

$$\mathcal{P}_t := f(\hat{\theta}_t | K, \mathcal{D}_t^*). \tag{1}$$

$\hat{\theta}_t$ refers to certain optimal parameter settings of the underlying recommender system at each timestamp t:

$$\hat{\theta}_t := \arg\min_{\theta_t \in \Theta} \sum_{z_i \in \mathcal{D}_t^*} \mathcal{L}(z_i, \theta_t), \tag{2}$$

where Θ is the search space of model parameters, z_i is a user-item interaction from the current data segment, and $\mathcal{L}(z_i, \theta_t)$ can be any recommendation loss function at timestamp t to measure the disparity against the ground-truth. Our objective of Eqn. (1) thus is to consistently find the influential exemplars that further boost the recommendation performance for the future timestamp. A high-level workflow is depicted in Figure 2.

Vanilla Solution: Influence-based Replay

Specifically, to implement $f(\cdot)$ in Eqn. (1) for sampling the most influential replay data, we aim to understand the dependence of recommender model predictions on a *single user-item interaction*. While a straightforward manner would be *leave-one-out retraining* (Basu, You, and Feizi 2020), which is however computationally infeasible, we incorporate *influence function* (Koh and Liang 2017) to avoid this inadequacy. Generally, the influence function is an important concept in Robust Statistics, which measures the effect of a change in one observation on an estimator. To fill in the reservoir at this timestamp t, i.e., $\mathcal{P}_t$, for any interaction candidate $z_m \in \mathcal{D}_t^*$, our initial idea is to explicitly observe the impact of upweighting z_m by some small perturbation ϵ. This produces the following new parameter settings:

$$\hat{\theta}_{t,\epsilon,z_m} := \arg\min_{\theta_t \in \Theta} \big(\epsilon \mathcal{L}(z_m, \theta_t) + \sum_{z_i \in \mathcal{D}_t^*} \mathcal{L}(z_i, \theta_t)\big). \tag{3}$$

Under the strict *convexity* and *second-order differentiability* of the loss function $\mathcal{L}$, the *influence* of upweighting z_m on parameters θ_t, i.e., achieving $\hat{\theta}_{t,\epsilon,z_m}$ rather than $\hat{\theta}_t$, denoted by $\mathcal{I}_{\text{params}}(z_m)$, can be quantified with the first-order Taylor's approximation (Scanlon 1996; Koh and Liang 2017):

$$\mathcal{I}_{\text{params}}(z_m) := \frac{d\hat{\theta}_{t,\epsilon,z_m}}{d\epsilon}\bigg|_{\epsilon=0} = -H_{\hat{\theta}_t}^{-1} \nabla_{\theta_t} \mathcal{L}(z_m, \hat{\theta}_t), \tag{4}$$

where ∇ is the first-order differential operator and $H_{\hat{\theta}_t} = \sum_{z_i \in \mathcal{D}_t^*} \nabla_{\theta_t}^2 \mathcal{L}(z_i, \hat{\theta}_t)$ denotes the Hessian matrix and is positive definite by assumption.

While the original definition, i.e., Eqn. (4), measures z_m's influence on the model parameters, in incremental learning for recommendation, we are however more interested in *measuring z_m's influence on recommendation predictions*. Ideally, such influence on recommendation results should be measured as completely as possible, which can be formulated as follows:

$$\mathcal{I}_{\text{rec\_loss}}(z_m) := \sum_{z_i \in \mathcal{D}_{0:t}} \frac{d\mathcal{L}(z_i, \hat{\theta}_{t,\epsilon,z_m})}{d\epsilon}\bigg|_{\epsilon=0}. \tag{5}$$

We use notation $\mathcal{D}_{0:t}$ to denote all historical interactions. Based on the computed values $\mathcal{I}_{\text{rec\_loss}}$ from $\mathcal{D}_{0:t}$, we hope to retrieve the most influential samples such that they make substantial impacts on all previous data. This actually implies that the retrieved samples are representative to maintain the earlier learned knowledge such as users' long-term preferences. Noteworthily, in the incremental learning setting, the complete historical data before timestamp t, i.e., $\mathcal{D}_{0:t-1}$, is actually unavailable. We thus use the empirical

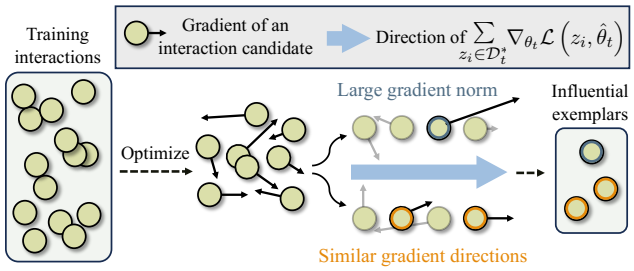


Figure 3: Influential exemplar retrieval of INFER$_{\text{ONCE}}$.

risk on $\mathcal{P}_{t-1}$ to approximate the terms derived by the original $\mathcal{D}_{0:t-1}$, by presuming a close correlation between them. Hence, Eqn. (5) is rewritten as follows:

$$\begin{aligned}\mathcal{I}_{\text{rec\_loss}}(z_m) &\approx \sum_{z_i \in \mathcal{P}_{t-1} \cup \mathcal{D}_t} \frac{d\mathcal{L}(z_i, \hat{\theta}_{t,\epsilon,z_m})}{d\epsilon}\Big|_{\epsilon=0} \\ &= \sum_{z_i \in \mathcal{D}_t^*} \nabla_{\theta_t} \mathcal{L}(z_i, \hat{\theta}_t)^\top \cdot \mathcal{I}_{\text{params}}(z_m).\end{aligned} \tag{6}$$

To keep reservoir $\mathcal{P}$ informative, we thus pick out those interactions with the most K negative numerical values, which conversely indicates the most representative data for recommender models to capture users' long-term preferences.

INFER$_{\text{ONCE}}$: One-step Influence Estimation

Limitations of Vanilla Implementation. However, the vanilla implementation may exhibit practical flaws. Our discussions are twofold: (1) Intuitively, inverting the Hessian matrix in $\mathcal{I}_{\text{params}}$ term of Eqn. (6) is computationally intensive, i.e., quadratic to the size of parameters θ_t of the downstream recommender backbone. This is unacceptable as modern recommender systems typically have hundreds of millions of parameters, particularly for scenarios with limited computation resources. We detail the time complexity analysis in the later section. (2) Moreover, as we will empirically elaborate in experiments, such vanilla version may underperform for large recommender models, which is also observed by recent research (Basu, Pope, and Feizi 2021; Epifano et al. 2023).

Fast Influence Estimation. We now formally introduce our advanced method **O**ne-step i**N**fluen**C**e **E**stimation (INFER$_{\text{ONCE}}$). Recall in Eqn. (3), to algorithmically compute $\hat{\theta}_{t,\epsilon,z_m}$, one may adopt gradient-based methods, e.g., Stochastic Gradient Descent (SGD), to conduct *iterative* optimization until convergence:

$$\theta_t^{(h+1)} \leftarrow \theta_t^{(h)} - \eta \nabla_{\theta_t}(\epsilon \mathcal{L}(z_m, \theta_t^{(h)}) + \sum_{z_i \in \mathcal{D}_t^*} \mathcal{L}(z_i, \theta_t^{(h)})). \tag{7}$$

With a slight notation abuse, we use $\theta_t^{(h)}$ to specifically refer to the intermediate status of θ_t at iteration h. Here η denotes the learning rate. Then $\hat{\theta}_{t,\epsilon,z_m}$ is defined as the converged value of θ_t from Eqn. (7). Since the efficiency bottleneck of vanilla implementation lies in the exhaustive iterations, i.e., to implement Eqn. (7), we thus propose to approximate $\hat{\theta}_{t,\epsilon,z_m}$ with only one-step estimation as follows:

$$\hat{\theta}_{t,\epsilon,z_m} := \theta_t - \lambda \nabla_{\theta_t}(\epsilon \mathcal{L}(z_m, \theta_t) + \sum_{z_i \in \mathcal{D}_t^*} \mathcal{L}(z_i, \theta_t)). \tag{8}$$

Which significantly reduces the computation cost compared to Eqn. (7) whilst providing flexibility of tuning approximation rate with hyper-parameter λ. Then we derive our one-step influence estimation, denoted by $\mathcal{I}^*_{\text{rec\_loss}}(\cdot)$, to similarly approximate the vanilla one in Eqn. (6) as follows:

$$\begin{aligned}\mathcal{I}^*_{\text{rec\_loss}}(z_m) &:= \sum_{z_i \in \mathcal{D}_t^*} \frac{d\mathcal{L}(z_i, \hat{\theta}_{t,\epsilon,z_m})}{d\epsilon}\Big|_{\epsilon=0} \\ &= \sum_{z_i \in \mathcal{D}_t^*} \nabla_{\theta_t} \mathcal{L}(z_i, \hat{\theta}_t)^\top \cdot \frac{d\hat{\theta}_{t,\epsilon,z_m}}{d\epsilon}\Big|_{\epsilon=0} \\ &= -\lambda \sum_{z_i \in \mathcal{D}_t^*} \nabla_{\theta_t} \mathcal{L}(z_i, \hat{\theta}_t)^\top \cdot \nabla_{\theta_t} \mathcal{L}(z_m, \theta_t).\end{aligned} \tag{9}$$

We observe from Eqn.(9) that, for any candidate z_m throughout the whole training data $\mathcal{D}_t^*$, its estimated influence $\mathcal{I}^*_{\text{rec\_loss}}(z_m)$ is determined as significant by two manners:

- either, the gradient norm of z_m at parameters θ_t, i.e., $||\nabla_{\theta_t} \mathcal{L}(z_m, \theta_t)||_2$, is as large as possible that reflects a substantial marginal impact to the downstream recommendation loss;
- or, the gradient direction of $\nabla_{\theta_t} \mathcal{L}(z_m, \theta_t)$ is as consistent as possible to the direction of term $\sum_{z_i \in \mathcal{D}_t^*} \nabla_{\theta_t} \mathcal{L}(z_i, \hat{\theta}_t)$ that regards to the recommendation losses capturing early model knowledge for all users in $\mathcal{D}_t^*$.

Therefore, these influential exemplars produced by our INFER$_{\text{ONCE}}$ are also in line with our intuition. We provide a visual illustration in Figure 3.

Approximation Discussion of INFER$_{\text{ONCE}}$. During model training at each timestamp t, we encourage θ_t to approach $\hat{\theta}_t$. Then the estimated influence in Eqn. (9) is similar to the vanilla one in Eqn. (6), only excluding the inverse Hessian matrix $H^{-1}_{\hat{\theta}_t}$. Notice that this covariant-weighted matrix measures the inter-dependence between the training samples, as it gauges their "resistance" to other samples' perturbations. In recommendation scenarios, such inter-dependence in underlying interaction data tends to be complex and noisy, due to the subjective nature of user behaviors and the sensitivity to external factors. As for our INFER$_{\text{ONCE}}$ methodology, we thus omit those high-order gradients that are extremely expensive to analytically calculate or empirically observe. This provides a straightforward yet effective way to measure the influence of a single interaction on model optimization, thus expediting the convergence with less complexity. We demonstrate this later in runtime experiments of Section 11. The pseudocodes of INFER$_{\text{ONCE}}$ are attached in Algorithm 1.

Complexity Analysis. The time complexity of our vanilla method INFER$_{\text{Vanilla}}$ in Eqn. (6) is $O(|\mathcal{D}_t^*|q^2 + q^3)$, where q is the total number of model parameters and variables. On the contrary, the time complexity of our advanced solution INFER$_{\text{ONCE}}$ in Eqn. (9) is $O(|\mathcal{D}_t^*|q)$ with $q > |\mathcal{D}_t^*|$.

Algorithm 1: Working Procedure of INFER$_{\text{ONCE}}$

Input: the sequence of user-item interaction segments $\mathbb{D} = \{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_T\}$; influential exemplar number K; model parameters $\theta_{\{0,\cdots,T\}}$ of the underlying recommender backbone $RecSys$.

$\theta_0 \leftarrow$ Initialization;
for $t = 0$ *to* T **do**
  $\mathcal{P}_t \leftarrow \emptyset$;
  if $t = 0$ **then**
    $\mathcal{D}_0^* \leftarrow \mathcal{D}_0$;
  else
    $\mathcal{D}_t^* \leftarrow \mathcal{D}_t \cup \mathcal{P}_{t-1}$;
  Training $RecSys$ with $\mathcal{D}_t^*$ starting from setting θ_t;
  $\hat{\theta}_t \leftarrow$ Optimal parameters of well-trained $RecSys$;
  $\mathcal{P}_t \leftarrow f(\hat{\theta}_t | K, \mathcal{D}_t^*)$ with INFER$_{\text{ONCE}}$;
  $\theta_{t+1} \leftarrow \hat{\theta}_t$;

Experiments

In this section, we present a series of empirical evaluations with the aim of addressing the following research questions:

- **RQ1:** How does INFER$_{\text{ONCE}}$ plug-and-play for different backbone models and perform compared to other incremental methodologies?
- **RQ2:** How close is INFER$_{\text{ONCE}}$ to the *"ideally-optimal"* case of model retraining with complete historical data?
- **RQ3:** Can INFER$_{\text{ONCE}}$ possibly further enhance end-to-end incremental recommender models?
- **RQ4:** How does INFER$_{\text{ONCE}}$ optimize over the vanilla implementation in the recommendation scenario?
- **RQ5:** How does the reservoir size of INFER$_{\text{ONCE}}$ influence the model performance?

Experimental Setups

Evaluation Benchmarks. We incorporate four real-world benchmarks that vary in *size*, *domain*, *sparsity*, and *duration*, as reported in Table 1. Each dataset is partitioned chronologically into a 50% *base segment* and five consecutive 10% *incremental segments*. The base segment is randomly divided into training, validation, and testing sets in a 6:2:2 ratio. For incremental segments, to imitate real production scenarios, we retrieve recommender models trained in the previous step and assess their performance with the most recent consecutive data. The data is evenly divided into two halves, with one half used for validation and the other for testing.

Evaluation Metrics. We consider two evaluation tasks: Click-through rate (CTR) prediction and Top-N recommendation. CTR prediction is evaluated using *AUC*, while Top-N recommendation uses *Recall@*N. For stable reproducibility, we conduct five-fold cross validation. We report testing results averaged on all incremental updates.

Backbone Models. To demonstrate the *generalization* of INFER$_{\text{ONCE}}$, we implement it into four renowned neural recommender models: DeepFM (Guo et al. 2017) and TwoTower (Huang et al. 2013) for CTR prediction, MF-BPR (Rendle et al. 2012) and LightGCN (He et al. 2020) for Top-N recommendation.

	Lastfm-2k	TB2014	Gowalla	Foursquare
# Users	1,090	8,844	29,858	51,919
# Items	3,646	39,103	40,988	37,320
# IR (M)	0.05	0.75	1.03	2.35
# S (m)	13.22	2.17	0.84	1.21
# Time	56yrs	31dys	19mths	22mths

Table 1: Dataset statistics. # IR (M) represents the number of interactions in millions (10^6). # S (m) refers to the sparsity in thousandths (10^{-3}).

Competing Algorithms. Apart from our basic implementation, INFER$_{\text{Vanilla}}$, we additionally include eight competing methods that are broadly subsumed into three groups.

- **Conventional Implementations**: (1) Fine-Tune can be considered as the *lower-bound* of incremental models which just fine-tunes the previously trained model solely on the incremental segment. (2) FD-Retrain entails exhaustive full-data retraining at each time of data update. It can be considered the *upper-bound* performance based on all the accumulated data access.
- **Experience Replay Methods**: (3) Uniform (Vitter 1985; Ahrabian et al. 2021) is a classic *experience replay* strategy that uniformly samples a subset of historical data as the incremental segment. (4) ER-MIR (Rahaf and Lucas 2019) predicts parameter updates to retrieve the pivotal samples. (5) InvDeg (Ahrabian et al. 2021), taking inspiration from Gdumb (Prabhu, Torr, and Dokania 2020), leverages the graph properties for sampling. We further extend it to the non-graph-based scenario. For a fair comparison, we keep the same reservoir size for replay.
- **Incremental Recommender Models**: (6) GraphSAIL (Xu et al. 2020), (7) SGCT (Wang, Zhang, and Coates 2021), and (8) LWC-KD (Wang, Zhang, and Coates 2021) are three state-of-the-art methods that based on graph neural networks. They employ miscellaneous designs such as knowledge distillation mechanisms to preserve key model knowledge that was learned from historical data.

Overall Performance (RQ1)

In this section, we present an overall performance analysis between INFER$_{\text{ONCE}}$ and all baselines for tasks of CTR prediction and Top-N recommendation. The results are shown in Table 2, and our empirical observations are twofold:

- **CTR prediction:** (1) Our method outperforms Fine-Tune implementation across all backbone models and datasets, yielding performance improvements from 0.22% to 5.12% in terms of AUC. This highlights the efficacy of INFER$_{\text{ONCE}}$ in mitigating the forgetting tendency that may occur when simply fine-tuning a previously trained model. (2) In comparison to other IL methods, INFER$_{\text{ONCE}}$ consistently exhibits superior performance on CTR prediction. The only exception is the experiment on Foursquare using DeepFM as the backbone model.

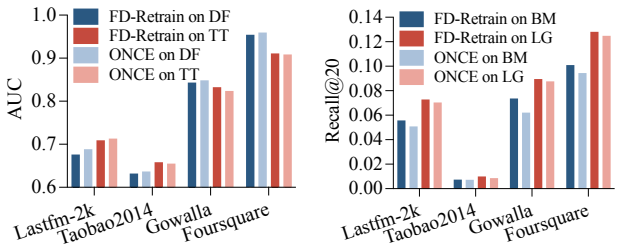
Task	Backbone / Method	Lastfm-2k		Taobao2014		Gowalla		Foursquare	
		DeepFM	TwoTower	DeepFM	TwoTower	DeepFM	TwoTower	DeepFM	TwoTower
CTR	Fine-Tune	0.6858	0.7054	0.6356	0.6472	0.8403	0.7836	0.9569	0.9000
	Uniform	0.6814	0.7091	0.6367	0.6495	0.8450	0.8093	0.9585	0.9057
	ER-MIR	0.6823	0.7097	0.6330	0.6510	0.8441	0.8174	0.9587	0.9054
	InvDeg	0.6766	0.6951	0.6351	0.6502	0.8448	0.8035	**0.9598**	0.8993
	$\text{INFER}_{\text{Vanilla}}$	0.6863	0.7099	0.6338	0.6528	0.8432	0.8052	0.9547	0.9062
	$\text{INFER}_{\text{ONCE}}$	**0.6884**	**0.7134**	**0.6370**	**0.6551**	**0.8486**	**0.8237**	0.9595	**0.9087**
		BPR-MF	LightGCN	BPR-MF	LightGCN	BPR-MF	LightGCN	BPR-MF	LightGCN
Top-N	Fine-Tune	0.0450	0.0479	0.0066	0.0073	0.0573	0.0752	0.0842	0.1074
	Uniform	0.0469	0.0660	0.0063	0.0081	0.0588	0.0838	0.0913	0.1189
	ER-MIR	0.0475	0.0668	0.0069	0.0076	0.0587	0.0823	0.0908	0.1089
	InvDeg	0.0479	0.0655	0.0067	0.0079	0.0602	0.0845	0.0922	0.1200
	GraphSAIL	-	0.0553	-	0.0078	-	0.0725	-	0.1042
	SGCT	-	0.0620	-	0.0080	-	0.0783	-	0.1157
	LWC-KD	-	0.0674	-	0.0081	-	0.0792	-	0.1194
	$\text{INFER}_{\text{Vanilla}}$	**0.0513**	0.0695	**0.0074**	0.0083	0.0591	0.0847	0.0913	0.1163
	$\text{INFER}_{\text{ONCE}}$	0.0508	**0.0704**	0.0072	**0.0086**	**0.0621**	**0.0876**	**0.0944**	**0.1249**

Table 2: (1) Overall performance on tasks of CTR prediction (AUC) and Top-N recommendation (*Recall*@20) based on five-fold cross validation. (2) The best results are bold and the second-best values are underlined.

- **Top-N recommendation:** (1) While $\text{INFER}_{\text{Vanilla}}$ performs well on smaller datasets, i.e., Lastfm-2k and Taobao2014, it however downgrades on larger ones, i.e., Gowalla and Foursquare. This can be attributed that the vanilla influence function tends to be sensitive to the *scale* of network structures or model parameters (most recommender models require explicit size allocation for both user- and item-embeddings), while the lightweight is more favorable to produce more accurate and stable performance (Basu, Pope, and Feizi 2021). On the contrary, our $\text{INFER}_{\text{ONCE}}$ performs the best for larger datasets, which reaffirms its effectiveness for Top-N recommendation. (2) Moreover, compared to the latest graph-based models, i.e., GraphSAIL, SGCT, and LWC-KD, $\text{INFER}_{\text{ONCE}}$ is model-agnostic and can be adapted to the non-graph-based backbone models, e.g., BPR-MF. This demonstrates its generalization capability with consistent performance superiority. The RQ3 section empirically investigates how $\text{INFER}_{\text{ONCE}}$ further enhances these graph-based models.

$\text{INFER}_{\text{ONCE}}$ *v.s.* Full-data Retraining (RQ2)

We compare $\text{INFER}_{\text{ONCE}}$ with full-data retraining, which provides insights into how close our model is to the "optimal" case. We exhaustively repeat the experiments similar to the evaluation paradigm for RQ1 and plot the stacked-column charts in Figure 4. We notice that (1) for CTR prediction, $\text{INFER}_{\text{ONCE}}$ is competitive with even slightly better performance than retraining, e.g., plugged on DeepFM on Lastfm-2k, Gowalla, and Foursquare datasets. This showcases that $\text{INFER}_{\text{ONCE}}$ can well "memorize" users' preferences to make correct click identification for recommendations. (2) While the CTR task can be interpreted as binary classification, Top-N recommendation could be more difficult, as the *relative item ranking* personalized by each user is of the essence. Consequently, compared to model retraining with complete historical data, $\text{INFER}_{\text{ONCE}}$ is left with a discernible gap in terms of *Recall*@20 metric. However, one straightforward flaw is that full-data retraining is extremely expensive, we detail their runtime costs later.



(a) CTR Prediction (b) Top-N Recommendation

Figure 4: ONCE *v.s.* full-data retraining. Due to space limitations, DeepFM, TwoTower, BPR-MF, and LightGCN are denoted as DF, TT, BM, and LG respectively.

Cross-model Compatibility Analysis (RQ3)

As we mentioned earlier, three state-of-the-art graph-based models, i.e., GraphSAIL, SGCT, and LWC-KD, incorporate the *knowledge distillation* mechanism to reduce the disparity between their previously-trained and newly-updated versions. We then apply $\text{INFER}_{\text{ONCE}}$ to enhance them by providing influential exemplar replay and show the resultant *Recall*@20 metric in Table 3. Notably, the substantial performance improvements are reasonable as our replay strategy is technically orthogonal to their methodologies. More importantly, this indicates that $\text{INFER}_{\text{ONCE}}$ can thus synergistically boost their performance across all datasets.

Empirical Study on $\text{INFER}_{\text{ONCE}}$ (RQ4)

Runtime Efficiency. The first motivation of $\text{INFER}_{\text{ONCE}}$ to optimize $\text{INFER}_{\text{Vanilla}}$ is for computation acceleration. We showcase the holistic runtime costs of all models on the largest dataset Foursquare in Figure 5. We observe that: (1)

Method	Lastfm-2k	TB2014	Gowalla	Foursquare
GraphSAIL	0.0553	0.0078	0.0725	0.1042
+ INFER$_{\text{ONCE}}$	**0.0708**	**0.0082**	**0.0815**	**0.1195**
SGCT	0.0620	0.0080	0.0783	0.1157
+ INFER$_{\text{ONCE}}$	**0.0722**	**0.0088**	**0.0881**	**0.1284**
LWC-KD	0.0674	0.0081	0.0792	0.1194
+ INFER$_{\text{ONCE}}$	**0.0721**	**0.0088**	**0.0902**	**0.1292**

Table 3: *Recall*@20 of INFER$_{\text{ONCE}}$-enhanced models.

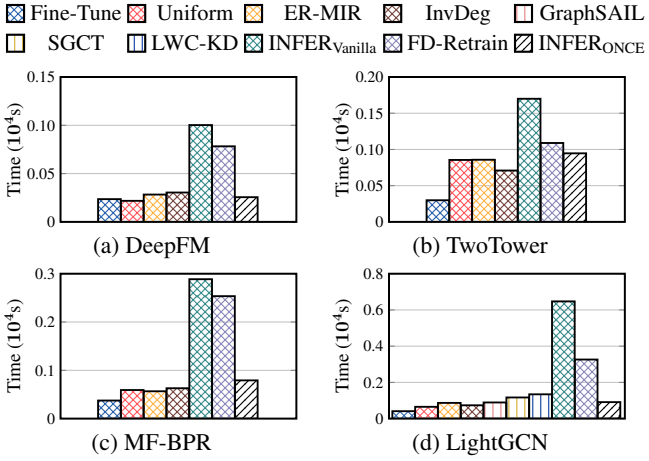


Figure 5: The holistic runtime cost (s) on the Foursquare dataset including reservoir construction, incremental model training with early-stop, and evaluation (best view in color).

INFER$_{\text{ONCE}}$ achieves much less runtime cost compared to INFER$_{\text{Vanilla}}$ whilst being on par with incremental learning methods, i.e., ER-MIR, InvDeg, and Uniform, across various backbone models. (2) Compared to recent incremental recommender models, e.g., GraphSAIL and LWC-KD, our method exhibits faster convergence speed, showing the efficiency of our one-step estimation design.

Comparison with INFER$_{\text{Vanilla}}$. To understand how INFER$_{\text{ONCE}}$ approximates INFER$_{\text{Vanilla}}$with different model parameter sizes, we visually compare their calculated results based on Lastfm-2k and Gowalla datasets. Specifically, for both of these implementations, we first respectively compute their influence values and rank them accordingly. Then we enlarge the recommender model size around 16 times, e.g., from 38k to 606k for Lastfm-2k dataset. For each dataset, we randomly sample 1,000 candidates and compare their ranking differences produced by INFER$_{\text{ONCE}}$ and INFER$_{\text{Vanilla}}$. Intuitively, for each data candidate, the smaller the ranking difference is, the more similar these two implementations would be. We plot the ranking difference distributions with kernel density estimation in Figure 6. We observe the following trends: (1) Irrespective of the backbone models' size, these two implementations exhibit substantial overlap in their distributions, indicating that INFER$_{\text{ONCE}}$ closely resembles INFER$_{\text{Vanilla}}$ with not many deviations. (2) With the model size increasing, we notice that their ranking disparities slightly rise in central distribution regions which corresponds to the case of minor differences in rankings. However, considering our early empirical findings, we believe that such small disagreements are acceptable as INFER$_{\text{ONCE}}$ essentially has more focus on efficiency consideration.

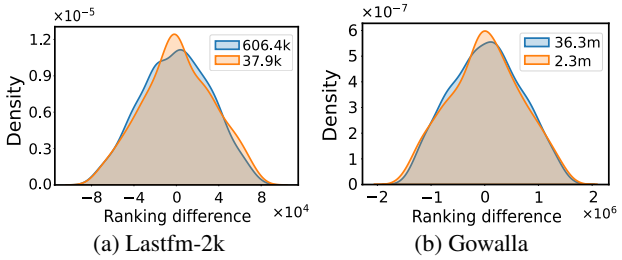


Figure 6: Distributions of ranking differences between our two implementations by differing model sizes.

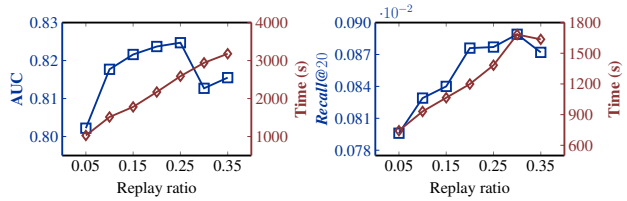


Figure 7: Results of varying replay ratios.

Parameter Study on Reservoir Size (RQ5)

Lastly, we vary the reservoir size by adjusting the reply ratio, i.e., $K/|\mathbb{D}|$, to investigate the INFER$_{\text{ONCE}}$ performance. The reply ratio is altered from 0.05 to 0.35 at intervals of 0.05 and we implement INFER$_{\text{ONCE}}$ to two backbones, i.e., TwoTower and LightGCN, on Gowalla dataset. As shown in Figure 7, the model performance continually improves with the increasing ratio from 0.05 to 0.25 (TwoTower) or 0.30 (LightGCN). However, after further expanding the reservoir, apart from the increasing time costs, we also notice performance degradation, e.g., dropping from 0.8247 to 0.8127 of AUC on TwoTower. The explanation is straightforward as over-training the historical data may introduce bias that is not descriptive of new data, which is reasonable in practice to reflect the user's preference drifting phenomenon.

Conclusion

In this work, we investigate incremental learning for recommender systems via influential exemplar replay. While our vanilla implementation finds the most analytically influential samples, our advanced solution INFER$_{\text{ONCE}}$ optimizes toward the efficiency bottleneck. Extensive empirical evaluation demonstrates not only the generalization of our methods across four backbone models but also the performance superiority over state-of-the-art incremental recommender models on four public benchmarks. For future work, we plan to explore influence functions that encapsulate graph structural information (Liu et al. 2022; Wu et al. 2022). Another promising direction is to devise generative models (Li et al. 2020; Ho, Jain, and Abbeel 2020) capable of synthesizing user interactions from historical archives, enhancing the practicality through generative replay.

Acknowledgments

We thank anonymous reviewers for their valuable comments. The work described here was partially supported by grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 14222922, RGC GRF No. 2151185) and (RGC Research Impact Fund R5034-18; CUHK 2410021). Chenhao Ma was supported in part by NSFC Grant (No. 62302421). Yixiang Fang was supported in part by NSFC Grant (No. 62102341).

References

Ahrabian, K.; Xu, Y.; Zhang, Y.; Wu, J.; Wang, Y.; and Coates, M. 2021. Structure aware experience replay for incremental learning in graph-based recommender systems. In *CIKM*, 2832–2836.

Aljundi, R.; Chakravarty, P.; and Tuytelaars, T. 2017. Expert gate: Lifelong learning with a network of experts. In *CVPR*, 3366–3375.

Basu, S.; Pope, P.; and Feizi, S. 2021. Influence functions in deep learning are fragile. In *ICLR*.

Basu, S.; You, X.; and Feizi, S. 2020. On second-order group influence functions for black-box predictions. In *ICML*, 715–724.

Bonicelli, L.; Boschini, M.; Porrello, A.; Spampinato, C.; and Calderara, S. 2022. On the effectiveness of lipschitz-driven rehearsal in continual learning. *NeurIPS*, 35: 31886–31901.

Chen, C.; Zhang, M.; Zhang, Y.; Liu, Y.; and Ma, S. 2020. Efficient neural matrix factorization without sampling for recommendation. *TOIS*, 38(2): 1–28.

Chen, Y.; Fang, Y.; Zhang, Y.; and King, I. 2023a. Bipartite Graph Convolutional Hashing for Effective and Efficient Top-N Search in Hamming Space. In *WWW*, 3164–3172.

Chen, Y.; Guo, H.; Zhang, Y.; Ma, C.; Tang, R.; Li, J.; and King, I. 2022a. Learning binarized graph representations with multi-faceted quantization reinforcement for top-k recommendation. In *SIGKDD*, 168–178.

Chen, Y.; Truong, T.; Shen, X.; Wang, M.; Li, J.; Chan, J.; and King, I. 2023b. Topological Representation Learning for E-commerce Shopping Behaviors. In *MLG-KDD*.

Chen, Y.; Yang, M.; Zhang, Y.; Zhao, M.; Meng, Z.; Hao, J.; and King, I. 2022b. Modeling scale-free graphs with hyperbolic geometry for knowledge-aware recommendation. In *WSDM*, 94–102.

Chen, Y.; Yang, Y.; Wang, Y.; Bai, J.; Song, X.; and King, I. 2022c. Attentive knowledge-aware graph convolutional networks with collaborative guidance for personalized recommendation. In *ICDE*, 299–311.

Chen, Y.; Zhang, Y.; Yang, M.; Song, Z.; Ma, C.; and King, I. 2023c. WSFE: Wasserstein Sub-graph Feature Encoder for Effective User Segmentation in Collaborative Filtering. In *SIGIR*, 2521–2525.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *RecSys*, 191–198.

De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2021. A continual learning survey: Defying forgetting in classification tasks. *TPAMI*, 44(7): 3366–3385.

Epifano, J. R.; Ramachandran, R. P.; Masino, A. J.; and Rasool, G. 2023. Revisiting the fragility of influence functions. *Neural Networks*, 162: 581–588.

Guo, H.; Tang, R.; Ye, Y.; Li, Z.; and He, X. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*.

Gupta, A.; Kafle, S.; Wen, D.; Wang, D.; Srivastava, S.; Sinha, S.; Gupta, N.; Jain, B.; Sankar, A.; and Zhang, L. 2020. Image and video understanding for recommendation and spam detection systems. In *SIGKDD*, 3577–3578.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NeurIPS*, 1025–1035.

He, B.; He, X.; Zhang, R.; Zhang, Y.; Tang, R.; and Ma, C. 2023a. Dynamic Embedding Size Search with Minimum Regret for Streaming Recommender System. In *CIKM*, 741–750.

He, B.; He, X.; Zhang, Y.; Tang, R.; and Ma, C. 2023b. Dynamically Expandable Graph Convolution for Streaming Recommendation. In *WWW*, 1457–1467.

He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, 639–648.

He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *WWW*, 173–182.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *NeurIPS*, 33: 6840–6851.

Huang, P.-S.; He, X.; Gao, J.; Deng, L.; Acero, A.; and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2333–2338.

Hung, C.-Y.; Tu, C.-H.; Wu, C.-E.; Chen, C.-H.; Chan, Y.-M.; and Chen, C.-S. 2019. Compacting, picking and growing for unforgetting continual learning. *NeurIPS*, 32.

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. *Proceedings of the national academy of sciences*, 114(13): 3521–3526.

Koh, P. W.; and Liang, P. 2017. Understanding black-box predictions via influence functions. In *ICML*, 1885–1894.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.

Li, J.; Li, Z.; Mou, L.; Jiang, X.; Lyu, M.; and King, I. 2020. Unsupervised text generation by learning from search. *NeurIPS*, 33: 10820–10831.

Li, Y.; Li, Z.; Wang, P.; Li, J.; Sun, X.; Cheng, H.; and Yu, J. X. 2023. A Survey of Graph Meets Large Language Model: Progress and Future Directions. *arXiv preprint arXiv:2311.12399*.

Liu, J.; Fournier-Viger, P.; Zhou, M.; He, G.; and Nouioua, M. 2022. CSPM: Discovering compressing stars in attributed graphs. *Information Sciences*, 611: 126–158.

Ma, H.; Zhou, D.; Liu, C.; Lyu, M. R.; and King, I. 2011. Recommender systems with social regularization. In *WSDM*, 287–296.

Ma, Y.; Song, Z.; Hu, X.; Li, J.; Zhang, Y.; and King, I. 2023. Graph Component Contrastive Learning for Concept Relatedness Estimation. *AAAI*, 37(11): 13362–13370.

Mallya, A.; and Lazebnik, S. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *CVPR*, 7765–7773.

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

Prabhu, A.; Torr, P. H.; and Dokania, P. K. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*, 524–540.

Rahaf, A.; and Lucas, C. 2019. Online Continual Learning with Maximally Interfered Retrieval. In *NeurIPS*.

Rannen, A.; Aljundi, R.; Blaschko, M. B.; and Tuytelaars, T. 2017. Encoder based lifelong learning. In *ICCV*, 1320–1328.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. icarl: Incremental classifier and representation learning. In *CVPR*, 2001–2010.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Scanlon, E. S. 1996. Residuals and influence in regression. *Insurance Mathematics and Economics*, 3(18): 228.

Song, Z.; Zhang, Y.; and King, I. 2023. Optimal Block-wise Asymmetric Graph Construction for Graph-based Semi-supervised Learning. In *NeurIPS*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph attention networks. *stat*, 1050(20): 10–48550.

Vitter, J. S. 1985. Random sampling with a reservoir. *TOMS*, 11(1): 37–57.

Wang, H.; Zhao, M.; Xie, X.; Li, W.; and Guo, M. 2019a. Knowledge graph convolutional networks for recommender systems. In *WWW*, 3307–3313.

Wang, X.; He, X.; Cao, Y.; Liu, M.; and Chua, T.-S. 2019b. Kgat: Knowledge graph attention network for recommendation. In *SIGKDD*, 950–958.

Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019c. Neural graph collaborative filtering. In *SIGIR*, 165–174.

Wang, Y.; Zhang, Y.; and Coates, M. 2021. Graph structure aware contrastive knowledge distillation for incremental learning in recommender systems. In *CIKM*, 3518–3522.

Wang, Z.; Shen, Y.; Zhang, Z.; and Lin, K. 2023. Feature staleness aware incremental learning for CTR prediction. In *IJCAI*.

Welling, M. 2009. Herding dynamical weights to learn. In *ICML*, 1121–1128.

Wu, Q.; Zhao, W.; Li, Z.; Wipf, D. P.; and Yan, J. 2022. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35: 27387–27401.

Xu, Y.; Zhang, Y.; Guo, W.; Guo, H.; Tang, R.; and Coates, M. 2020. Graphsail: Graph structure aware incremental learning for recommender systems. In *CIKM*, 2861–2868.

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.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Cui, L.; and Nguyen, Q. V. H. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *SIGIR*, 1294–1303.

Yuan, F.; Zhang, G.; Karatzoglou, A.; Jose, J.; Kong, B.; and Li, Y. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *SIGIR*, 696–705.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning through Synaptic Intelligence. In *ICML*, 3987–3995.

Zhang, X.; Chen, Y.; Gao, C.; Liao, Q.; Zhao, S.; and King, I. 2022a. Knowledge-aware Neural Networks with Personalized Feature Referencing for Cold-start Recommendation. *arXiv preprint arXiv:2209.13973*.

Zhang, Y.; Chen, Y.; Song, Z.; and King, I. 2023a. Contrastive Cross-scale Graph Knowledge Synergy. In *SIGKDD*.

Zhang, Y.; Zhu, H.; Chen, Y.; Song, Z.; Koniusz, P.; King, I.; et al. 2023b. Mitigating the Popularity Bias of Graph Collaborative Filtering: A Dimensional Collapse Perspective. In *NeurIPS*.

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I. 2022b. COSTA: covariance-preserving feature augmentation for graph contrastive learning. In *KDD*.

Zhang, Y.; Zhu, H.; Song, Z.; Koniusz, P.; and King, I. 2023c. Spectral feature augmentation for graph contrastive learning and beyond. In *AAAI*.

Zheng, G.; Zhang, F.; Zheng, Z.; Xiang, Y.; Yuan, N. J.; Xie, X.; and Li, Z. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *WWW*, 167–176.

Zhou, F.; and Cao, C. 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *AAAI*, volume 35, 4714–4722.