Feature Staleness Aware Incremental Learning for CTR Prediction

***Abstract*—Click-through Rate (CTR) prediction in real-world recommender systems often deal with billions of user interactions periodically. To improve the training efficiency, it is common to update the CTR prediction model incrementally using the new incremental data and a subset of historical data. However, the feature embeddings of a CTR prediction model often get stale when the corresponding features do not appear in current incremental data. In the next time span, the model would have a performance degradation on samples containing stale features, which we call the feature staleness problem. To mitigate this problem, we propose a Feature Staleness Aware Incremental Learning method for CTR prediction (FeSAIL) which adaptively replays samples containing stale features. We first introduce a staleness-aware sampling algorithm (SAS) to sample a fixed number of stale samples with high sampling efficiency. We then introduce a staleness-aware regularization mechanism (SAR) for a fine-grained control of the feature embedding updating. Furthermore, we enhance SAS and SAR to handle staleness imbalance for CTR prediction with multi-categorial feature fields. We instantiate FeSAIL with a general deep learning-based CTR prediction model and the experimental results demonstrate FeSAIL outperforms various state-of-the-art methods on four benchmark datasets.**

I. Introduction

Click-through Rate (CTR) prediction is to estimate the probability of a user clicking a recommended item in a specific context [1]–[6]. As real-world recommender systems often deal with billions of user interactions periodically, various methods [7]–[10] have been proposed to update CTR prediction model in an incremental manner, which means that model will be fine-tuned with newly arrived data and possibly with part of historical data.

Existing works on incremental learning for CTR prediction can be divided into sample-based methods and model-based methods. Sample-based methods [11]–[13] update model by replaying historical samples based on heuristics rules, such as prioritizing recency [12], [14], [15] and the extent of being forgotten [16], [17]. Model-based methods aim to transfer old knowledge learned from the past to the present model using knowledge distillation [18], [19] or meta learning [20], [21].

Typically, CTR prediction models [2], [12], [21], [22] involve a low-level feature embedding layer, followed by the high-level feature interaction and prediction layers. The parameters in the high-level layers are always updated with incremental data. However in the low-level layer, the feature embeddings will not be updated and get stale when the corresponding features do not appear in the incremental data [18]. We found there is a performance degradation on next time span's data containing these stale features, which we call the feature staleness problem. The feature staleness problem is quite prevalent among the mainstream incremental learning methods for training CTR prediction models. We use the real CTR prediction data Criteo [20] as an example which consists of 20 consecutive time spans' incremental datasets denoted as $\{\mathcal{D}_1, \mathcal{D}_2, \cdots \mathcal{D}_{20}\}$, where the feature value set of each incremental dataset is $\mathcal{F}_{D_t}$. We choose two state-of-the-art incremental learning methods, GAG [14] and ASMG [21], for illustration. In each time span t, we fine-tune the same base model with $\mathcal{D}_t$ by applying GAG or ASMG. We then test the model on **overall samples** in $\mathcal{D}_t$ and on the **stale samples** in $\mathcal{D}_t$ which contain stale feature $f \notin \mathcal{F}_{D_t}$. The average AUC performance on 11-20 time spans' datasets is reported in Figure 1. We can see that the models trained using GAG or ASMG report lower average AUC on stale samples than on overall samples, which indicates the models have forgotten the knowledge learned in stale features during the testing phase.

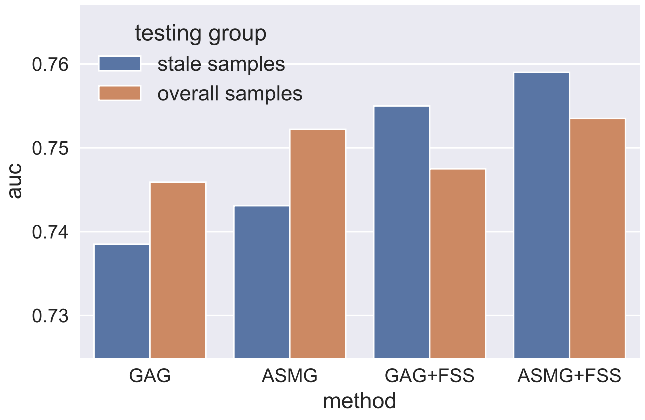


Fig. 1: The observed feature staleness problem on Criteo.

A vanilla way to mitigate the feature staleness problem is to replay all the samples containing stale features in each incremental step to ensure all features can be updated, which is referred as **Full stale sampling (FSS)**. **FSS** can serve as a replay plug-in for **GAG** and **ASMG**. As presented in Figure 1, **GAG+FSS** and **ASMG+FSS** can achieve remarkable AUC improvement on **stale samples**. However, this vanilla way has two limitations. First, in practice, the distribution of features can be scattered and vary among all incremental datasets. The set of all historical samples containing stale features will have an unaffordable size and randomly sampling a subset cannot guarantee a high coverage of stale features. Second, it treats all stale features in the same way regardless the extent of feature's staleness, which is not reasonable. For instance, repeatedly updating low-frequency features which keep stale for a long time will increase the risk of over-fitting [23]. On the contrary, some features only stale for one or two time spans, which are similar as unstale features, should be updated normally. In this paper, we propose a **Fe**ature **S**taleness **A**ware **I**ncremental **L**earning method for CTR prediction (FeSAIL) to overcome the above mentioned limitations. We first quantify the staleness of a feature and introduce a staleness-aware sampling algorithm (SAS) using a fixed-size reservior to store samples with stale features. We aim to cover as many features with small staleness values as possible, which is formulated as a weighted coverage problem and solved by a greedy algorithm. We then introduce a staleness-aware regularization mechanism (SAR) to restrict the updating of feature embeddings according to the extent of features' staleness.

The major contributions of this paper are summarized as follows.

- We observe the feature staleness problem existing in incremental learning of CTR prediction models and propose a feature staleness aware incremental learning method named FeSAIL to mitigate this problem. We introduce the SAS algorithm based on feature staleness to guarantee the sampling efficiency and design the SAR algorithm to restrict feature updating.
- Furthermore, we enhance SAS and SAR to handle staleness imbalance for CTR prediction with multi-categorial feature fields.
- We conduct extensive experiments and demonstrate that FeSAIL can efficiently alleviate the feature staleness problem and achieve 0.52% AUC improvement compared to various state-of-the-art methods on three widely-used public datasets and a private dataset with multi-categorial feature fields averagely.

II. Preliminaries

CTR prediction aims to estimate the probability that a user will click a recommended item. Generally, CTR models include three parts: embedding layer, interaction layer and prediction layer [2], [2], [18], [24], [25]. In typical CTR prediction tasks, the input of each sample is collected in a multi-field form [18], [23], [26]. The label y of each sample is 0 or 1 indicating whether a user clicked an item. Each field F_i $(1 \leq i \leq m)$ is filled with one specific feature f_i from feature space $\mathcal{F}_i$ with size n_i, where m is the number of fields. Each data sample is transformed into a dense vector via an embedding layer. Formally, for a sample $d = \{f_1, f_2, \cdots, f_m\}$, each feature is encoded as an embedding vector $e_i \in \mathcal{R}^{1\times k}$, where k is the embedding size. Therefore, each sample can be represented as an embedding list $E = (e_1^\top, e_2^\top, \cdots, e_m^\top) \in \mathcal{R}^{m\times k}$. Let the total number of features be $N = \sum_{i=1}^{m} n_i$. The embeddings of all the features form an embedding table $\mathcal{E} \in \mathcal{R}^{N\times k}$. In interaction layer, existing models [2], [23], [27], [28] utilize product operation and multi-layer perception to capture explicit and implicit feature interactions. And prediction $\hat{y}$ is generated as the probability of the user will click on a specific item. The cross-entropy loss is calculated between the label y and the prediction $\hat{y}$.

TABLE I: Notations

Notation	Description
y	label to indicate interaction
F_i	the i^{th} feature field
$\mathcal{F}_i$	the feature space of the i^{th} field
f_i	the specific feature filled in the i^{th} feature field
e_i	the embedding of feature in the i^{th} field
E	the embedding list to represent one sample
$\mathcal{E}$	the global embedding table
$\mathcal{D}_t$	the dataset in time span t
$\mathcal{R}_t$	the reservior in time span t

Suppose there is a sequence of CTR prediction datasets $\{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_t, \cdots\}$ indexed by time span. In incremental learning scenario, at time span t, $\mathcal{D}_t$ is the current dataset and $\mathcal{D}_{his}^t = \{\mathcal{D}_0, \mathcal{D}_1, \cdots, \mathcal{D}_{t-1}\}$ contains all the historical datasets. If we train the model on the current dataset and the entire historical datasets, the size of training data will grow overwhelmingly large. In many existing incremental learning methods for CTR prediction [11], [12], instead of preserving the whole historical datasets, they first initialize a reservior $\mathcal{R}_0$ using $\mathcal{D}_0$, which serves as a replay buffer. As time span t, they generate the reservior $\mathcal{R}_t$ based on $\mathcal{R}_{t-1} \cup \mathcal{D}_{t-1}$ and train the model using $\mathcal{R}_t$ and $\mathcal{D}_t$, which can greatly reduce the size of the training data. In this paper, we follow this setting and investigate how to find an efficient way for sampling R_t from $\mathcal{R}_{t-1} \cup \mathcal{D}_{t-1}$ and update the model parameters using $\mathcal{R}_t$ and $\mathcal{D}_t$ such that the model can achieve better performance on $\mathcal{D}_{t+1}$ at each time span t.

III. The FeSAIL Approach

A. Overview

The FeSAIL is a feature staleness aware incremental learning method for training CTR prediction models to mitigate the feature staleness problem illustrated in introduction. As presented in Figure 2, it can be divided into sampling and training stages for each time span t: In sampling stage, the staleness aware sampling (SAS) strategy first uses a reservior with a fixed size to store samples containing stale features and uses a greedy algorithm to cover as much staleness features as possible. In training stage, the model will then be trained with the staleness aware regularization (SAR) which restricts the embedding updating of features according to the extent of features' staleness.

B. Staleness Aware Sampling (SAS)

In this section, we introduce how our method chooses samples for reservior based on the feature staleness. Our key insight is that resampling samples with stale features, which refer to the features not appearing in current dataset, can help mitigate the performance degradation on new samples involving stale features in as illustrated Figure 1. A naïve way

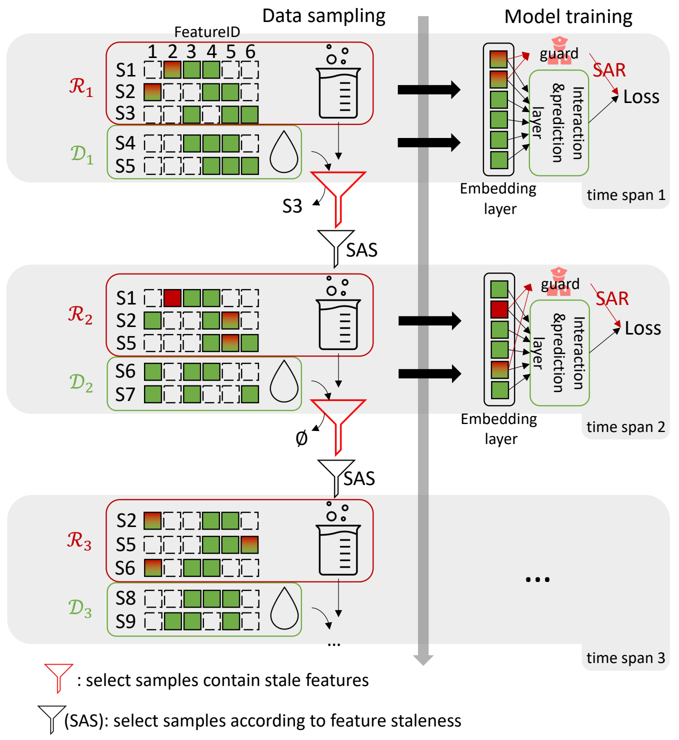


Fig. 2: The overview of FeSAIL. All features appeared in current dataset are in green square. The redness of a feature represents the extent of its staleness.

to generate $\mathcal{R}_t$ is to choose the samples from $\mathcal{R}_{t-1} \cup \mathcal{D}_{t-1}$ containing features that do not appear in $\mathcal{D}_t$, which is referred as **FSS**. More formally, let $\mathcal{F}_{D_t}$ denote the feature set of $\mathcal{D}_t$. We add a sample $d = \{f_1, f_2, \cdots, f_m\}$ in $\mathcal{R}_{t-1} \cup \mathcal{D}_{t-1}$ to $\mathcal{R}_t$ if d involves at least one feature that is not included in the feature set of $\mathcal{D}_t$. i.e., $\exists f_i \notin \mathcal{F}_{D_t}$ where $1 \leq i \leq m$. The model will be trained with $\mathcal{R}_t$ and $\mathcal{D}_t$ jointly. In this way, the chosen samples in $\mathcal{R}_t$ can supplement the feature set of $\mathcal{D}_t$ and update the feature embeddings such that the embeddings can adapt to the updated high-level layers in the model.

However in real scenarios such as news and short video recommendation, the distribution of features can be quite scattered and vary among each incremental datasets [20], [21]. Even using the above mentioned method will still preserve samples with an uncontrollable size and incur hight computation cost. Hence, a more desirable solution is to design a reservior with a fixed size L to store historical samples with stale features. Honestly, a fixed sized reservoir may not be able to cover all the stale features. To mitigate the performance degeneration to the maximal extent, we compute the fixed size reservior based on two rules. First, the designed reservior should cover as much stale features as possible. Second, the features with small staleness will be covered with higher priorities, because they might have higher possibilities to re-appear again in future datasets.

To solve this problem, we first quantify the feature staleness. Formally, each feature f_i is assigned with one staleness s_i^t which is initially set as 0. At time span t, the staleness of feature f_i will be:

$$s_i^t = \begin{cases} s_i^{t-1} + 1, & \text{if } f_i \notin \mathcal{F}_{D_t} \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

where $\mathcal{F}_{D_t}$ denotes the set of feature values contained in D_t.

Then we define the *weight* of each feature which has inverse correlation with its *staleness*:

$$w_i = func(s_i) + b, \tag{2}$$

where $func(\cdot)$ is an inverse correlation function like inverse proportional function or negative exponential function, and b is a bias term.

Our goal is to select a fixed number of samples which contain as much stale features as possible, which can be formulated as follows.

Definition 1 (Stale Features Sampling (SFS))**.** *Given the reservior $\mathcal{R} = \{d_1, d_2, \cdots, d_{|\mathcal{R}|}\}$ of **FSS** where each sample d_a contains N features $\{f_1, f_2, \cdots, f_N\}$ associated with weights $\{w_i\}_{i=1}^N$. The SFS problem is to find a collection of samples $\mathcal{R}^{fixed} \subseteq \mathcal{R}$, such that the total number of samples in $\mathcal{R}^{fixed}$ does not exceed a given capacity L and the total weight of features covered by $\mathcal{R}^{fixed}$ is maximized.*

The above problem is a generalized version of the standard maximum coverage problem (MCP) [29], which has been proved that an exact solution is intractable because the search space is too large. In this paper, we develop a greedy algorithm (SAS) which can yield an approximation ratio of $1 - \frac{1}{e}$ compared to the optimal solution.

Let W_a be the total weight of the features covered by sample d_a. Let W_a' denote the total weight of the features covered by set d_a, but not covered by any sample in $\mathcal{R}^{fixed}$. As presented in Algorithm 1, SAS totally has L iterations. In each iteration, SAS will calculate W_a' for each samples in $\mathcal{R}$ and select the sample with the maximum W_a'.

Algorithm 1: SAS algorithm

```
Input  : the vanilla reservior $\mathcal{R} = \{d_1, d_2, \cdots, d_{|\mathcal{R}|}\}$,
         $n$ features $\{f_1, f_2, \cdots, f_N\}$,
         associated weights $\{w_i\}_{i=1}^N$,
         a given capacity $L$
Output: a collection of samples $\mathcal{R}^{fixed} \subseteq \mathcal{R}$
1 $\mathcal{R}^{fixed} \leftarrow \emptyset$;$U \leftarrow \mathcal{R}$;
2 for $L$ iterations do
3     select $d_a \in \mathcal{R}$ that maximize $W_a'$;
4     if $U \neq \emptyset$ then
5         $\mathcal{R}^{fixed} \leftarrow \mathcal{R}^{fixed} \cup d_a$;
6         $U \leftarrow U \backslash d_a$;
7     end
8 end
```

Let b_a denote the total weight till ath iteration, i.e., $b_a = \sum_{j=1}^{a} W_j'$ and OPT denote the optimal solution of the SFSP; and c_a denotes the left weight for optimal, i.e., $c_a = OPT - b_a$. And let $b_0 = 0$, $c_0 = OPT$.

Lemma 1. *We have $W'_{a+1} \geq \frac{c_a}{L}$.*

Proof. Optimal solution reaches OPT weight at L iterations. That means, at each iteration $a+1$ there should be some samples in U whose contained weight greater than $\frac{c_a}{L-a} \geq \frac{c_a}{L}$. Otherwise, it was impossible to cover OPT weight at L steps by the optimal solution. Since the approximation algorithm is a greedy algorithm, i.e., choosing always the set covering maximum weight of uncovered elements, the weight of chosen set at each iteration should be at least the $\frac{1}{L}$ of the weight of remaining uncovered elements. $\square$

Lemma 2. *We have $c_{a+1} \leq (1-\frac{1}{L})^{a+1}) \cdot OPT$.*

Proof. We proof Lemma 2 by mathematical induction.
Base case: If $a = 0$,
we have $OPT - c_1 = b_1 = W'_1 \geq \frac{c_1}{L}$.
So $c_1 \leq (1-\frac{1}{L}) \cdot OPT$ and the lemma holds when $a = 0$.
Inductive hypothesis: Suppose the theorem holds for $a \geq 0$.
Inductive step: $c_a \leq (1-\frac{1}{L})^a \cdot OPT$
$c_{a+1} = c_a - W'_{a+1} \rightarrow c_{a+1} \leq c_a - \frac{c_a}{L}$
$\rightarrow c_{a+1} \leq (1-\frac{1}{L})^{a+1} \cdot OPT$.
So the theorem holds for $a+1$. $\square$

Theorem 1. *SAS can achieve an $1-\frac{1}{e}$ approximation ratio compared to the optimal solution for SFS.*

Proof. According to Lemma 2, we have $c_L \leq (1-\frac{1}{L})^L \cdot OPT = \frac{1}{e}OPT$.
so $b_L = OPT - c_L \geq OPT(1-\frac{1}{e})$. $\square$

From Theorem 1, we know that SAS will drop stale features for about 30% more than the optimal solution. However the SAS chooses the features with small staleness in prior and the actual drop ratio of stale features will be about 15%, which is testified in Section IV-E. The complexity of SAS is $O(|\mathcal{R}| * L * m)$, where m is the number of feature fields which still have high computation cost because both $|\mathcal{R}|$ and L can be million scale. Here we propose an optimized version called neighbor-based SAS which can decrease the complexity to $O(|\mathcal{R}| * m + |\bar{Q}|_{\mathcal{N}} * L * m)$, where the neighbors $\mathcal{N}$ refer to the samples which have at least one common stale features with the current chosen sample and $|\bar{Q}|_{\mathcal{N}}$ denotes the mean of the neighbor numbers which is in much smaller scale. Before Line 3 of Algorithm 1, we only need to update W'_a of these neighbor samples rather than all samples in the reservior.

C. Staleness Aware Regularization (SAR)

Intuitively, the features keeping stale for a long time are probably low-frequency features. Updating their embeddings too frequently will increase the risk of over-fitting [23], [30]. We follow the regularization-based methods [12], [18] to restrict the embedding updating of these kind of features using regularization term. However, a certain ratio of features only stale for one or two time spans, whose regularization term will be quite large and become an unaffordable burden for the model training. Thus it is unreasonable to treat all stale features equally regardless the extent of feature staleness.

TABLE II: The sample weight, feature number and staleness feature number for each field average on all samples of the Media dataset, sorted by sample weight in descending order.

Field id	Field weight	#Feature	#Staleness feature
352	13.1	15.8	11.2
318	6.1	14.2	4.9
317	5.9	14.5	4.4
...	...	...	...
Last 200+ fields	0	1	0

More formally, we add a regularization loss for feature f_i which called *guard*:

$$g_i = \frac{max(s_i, \eta)}{max(s_{max}, \eta)} \Delta e_i, \tag{3}$$

where s_{max} is the max staleness in current time span, Δe_i is embedding change for feature f_i between two consecutive mini-batches, and η is a hyperparameter used to control the scale of staleness. s_{max} is used to normalize the loss to prevent it ruining original model training. Thus the total loss of FeSAIL will be:

$$\mathcal{L} = \sum_{a \in \mathcal{D}_t + \mathcal{R}_t} (\mathcal{L}_{CE,a} + \lambda \sum_{i=0}^{N} g_i), \tag{4}$$

where $\mathcal{L}_{CE,a}$ refers to the cross entropy loss of model training on sample a, and λ is the regularization coefficient.

D. Handling Staleness Imbalance for Multi-Categorial Fields

The above SAS and SAR algorithms are mainly applicable to CTR data with single-categorical feature fields, i.e., there are only one feature in each field. In practice, CTR data may include multi-categorical feature fields which have multiple features in one field. SAS will sum up all the weights of the staleness feature as the total weight of this field (field weight), and the sample weight will be the sum of all field weights. Consequently, multi-categorial feature fields that have more features would have larger field weights and dominate in the final sample weight. To illustrate this phenomenon, we present the sample weight, feature number and staleness feature number for each field averaged on all samples of our private Media in Table II. The the top three fields dominate the total sample weight while a large percentage of fields are not considered by SAS, which is dubbed as the staleness feature imbalance.The degree of the imbalance becomes more significant for sparse fields like "Field 352" which involves a large number of stale features. This issue also exists in SAR, where the top fields will also dominate the regularization term and let the remaining fields neglected.

Thus, we propose to refine the SAS and SAR in a field-wise manner. The field weight of SAS will be calculated as:

$$w_{F_i} = \phi_{f_j \in F_i}(w_j). \tag{5}$$

We obtain the total sample weight W_a will be $\sum_i^m w_{F_i}$. Similarly, we replace the feature-wise guard of SAR as **field-wise guard** in Eq. (4):

$$g_{F_i} = \psi_{f_j \in F_i}(g_j). \tag{6}$$

We implement ϕ and ψ as mean or max function in the experiments. Note that the single-categorial feature fields can be regarded as special cases of multi-categorial feature fields and handled by field-wise SAS and SAR compatibly.

E. Implementation Details

We implemented our FeSAIL approach based using Pytorch 1.10 on a 64-bit Linus server equipped with 32 Intel Xeon@2.10GHz CPUs, 128GB memory and four RTX 2080ti GPUs. The proposed FeSAIL is model-agnostic. To test its effectiveness, we instantiate it on a general deep learning-based Embedding&MLP model, a network architecture that most of the CTR prediction models developed in recent years are based on [12], [18], [20], [21], [31]. The model mainly comprises embedding layers that transform high-dimensional sparse features into low-dimensional dense vectors, and Multi-Layer Perceptron (MLP) layers that learn the interaction of the concatenated feature embeddings. By default, we set the inverse correlation function in Eq. (2) as inverse proportional function and the bias term as 1. The ϕ and ψ in Eq. (5) and Eq. (6) are set as max function. The sampling reservior size L is set as the same size of the corresponding incremental datasets. We report results based on a grid search over the hidden layer size, initial learning rate and number of cross layers. The batch size is set from 256 to 4096. The embedding size and hidden layer sizes is chosen from 32 to 1024. For DCN, the number of cross layers is from 1 to 6. The η in 3 is from 5 to 10. We choose the Adam optimizer [32] to train the model with learning rate from 0.0001 to 0.001, and perform early stopping in the training process.

F. Complexity Analysis

1) Time Complexity: The computational complexity of SAS is $O(|\mathcal{R}| * m + |\bar{Q}| * L * m)$ using neighbor based optimization. The regularization term for SAR has complexity as $O(||\mathcal{R} + \mathcal{D}| * m)$. Thus the total external time complexity will be negligible compared to model training.

2) Space Complexity: SAS needs to store $|\mathcal{R}|$ samples, the size of which is fixed and can be set according to the actual memory space. And the SAR is a non-parameterized module which does not require extra storage space.

IV. Experiment

We conduct the experiments to evaluate the performance of our proposed method and answer the following questions :

- **RQ1** Can the proposed FeSAIL outperform the existing incremental learning methods on CTR prediction?
- **RQ2** How do the SAR and SAS algorithm affect the effectiveness of FeSAIL ?
- **RQ3** How does different inverse correlation functions and biases influence the performance of FeSAIL?
- **RQ4** How does SAS deal with features with different staleness and multi-categorial feature fields?

TABLE III: The statistics of the datasets.

dataset	#samples	#fields	#features
Criteo	10,692,219	26	1,849,038
iPinYou	21,920,191	21	4,893,029
Avazu	401,839,101	20	10,922,019
Media	104,416,327	337	60,833,522

A. Experimental Setups

1) Dataset: We use three real-world datasets from Criteo[1], iPinYou[2], Avazu[3] and one private industrial dataset collected from a commercial media platform.

- **Criteo** This dataset consists of 24 days' consecutive traffic logs from Criteo, including 26 categorical features, and the first column as label indicating whether the ad has been clicked or not.
- **iPinYou** This dataset is a public real-world display ad dataset with each ad display information and corresponding user click feedback. The data logs are organized by different advertisers and in a row-per-record format. There are 21.92M data instances with 14.89K positive label (click) in total. The features for each data instance are all categorical.
- **Avazu** This dataset contains users' click behaviors on displayed mobile ads. There are 23 feature fields including user/device features and ad attributes. The fields are partial anonymous.
- **Media** This is a real CTR prediction dataset collected from a commercial media platform. We extract 48 consecutive hours' data from April 23,2022, with 232 single-categorial feature fields and 115 multi-categorial feature fields, which containing more than one features like tags and user categories.

The detailed statistics of the datasets are listed in Table III. For fair comparison, we follow the same data preprocessing and data splitting rule for all the compared methods.

We filter out all features which appear in fewer than 10 samples as an "unknown" feature. For each dataset, the samples in the first half in time sequence serve as the pretraining datasets. And the remaining samples will be divided equally into 10 incremental datasets. We use the samples in last time span's incremental dataset and reservior to fine-tune the model and evaluate it on the next incremental dataset.

2) Evaluation Metrics: We use the following two metrics for performance evaluation: AUC (Area Under the ROC curve) and Logloss (cross entropy), which are commonly used in CTR prediction works [33], [34].

[1]https://www.criteo.com
[2]https://www.iPinYou.com
[3]https://www.kaggle.com/c/avazu-ctr-prediction

3) Baselines: We compare our proposed FeSAIL with existing two sample-based and two model-based incremental learning methods for training CTR prediction models.

- **Incremental Update (IU)** updates the model incrementally using only the new data.
- **Random Sample (RS)** updates the model using the new data and the randomly sampled historical data.
- **RMFX** [35] is a matrix factorization based model, which uses active learning sampling algorithm to help replay the historical samples.
- **GAG** [14] is a method which uses global attributed graph neural network for streaming session-based recommendation. We regard one sample in CTR task as a piece of session in this model.
- **EWC** [36] is a general incremental learning method which prevents the model from changing the informative parameters in previous datasets.
- **ASMG** [21] is a meta-learning based incremental method for CTR prediction models, which uses a sequentialized meta-learner to generate model parameters in the next time span.

RMFX+SAR and **GAG+SAR** add the guard term in Eq. (3) on the loss function of original RMFX and GAG. Similarly, **EWC+SAS** and **ASMG+SAS** using our sample method to enhance the performance of EWC and ASMG, which is designed to testify the effectiveness of guard term.

B. Experimental Results: RQ1

Table IV presents the overall performance on three public datasets averaged over 5 runs. We have the following observations. First, RS performs slightly better than IU which reflects that the future samples also need the old knowledge preserved in historical samples. Second, RMFX+SAR and GAG+SAR performs better than RMFX and GAG respectively, which shows restricting the feature embedding updating according to its staleness can also improve the performance of methods with other sampling strategies. Third, EWC+SAS and ASMG+SAS performs better than EWC and ASMG respectively, which indicates using a fixed size reservior to replay stale features is compatible with these model based method using other regularization strategies. Fourth, FeSAIL achieves 0.71%, 1.04%, and 0.68% relative improvements compared to GAG+SAS, the best sample based baseline using our regularization strategy and achieves 0.35%, 0.68%, and 0.30% relative improvements compared to ASMG+SAS, the best model based baseline using our sampling strategy.

Table V presents the overall performance on Media, where we have similar observations as those on the public datasets. More specifically, FeSAIL achieves 0.71% relative improvement compared to GAG+SAS, the best sample based baseline using our regularization strategy and achieves 0.52% relative improvements compared to ASMG+SAS, the best model based baseline using our sampling strategy. Figure 3 shows the disentangled performance at each time span. We can see that the performance of all methods have similar trend on different datasets and FeSAIL ourperforms all the other methods in most time spans.

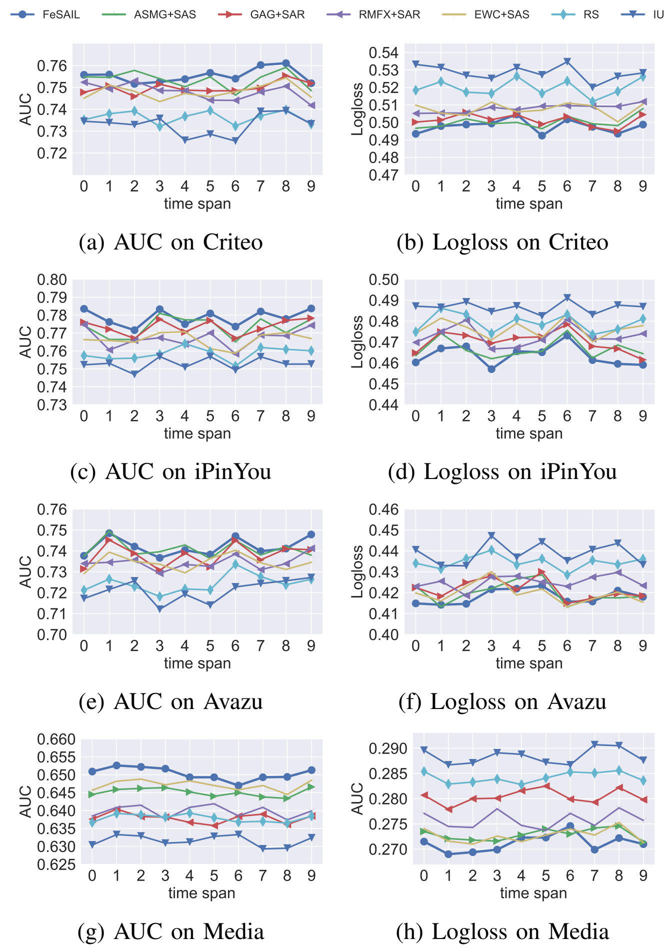


Fig. 3: Prediction performance on each time span. Note that we present six most competitive baselines and omit the remaining methods.

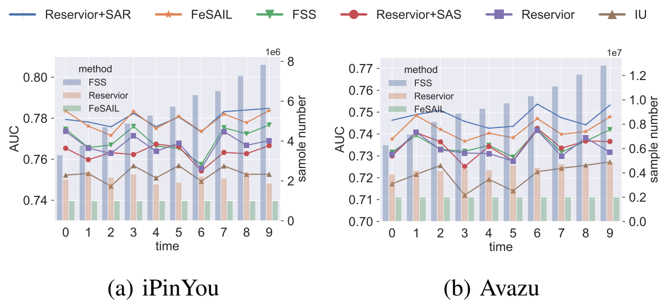


Fig. 4: Ablation study on iPinYou and Avazu.

C. Ablation Study: RQ2

1) Effects of SAS/SAR: We perform an ablation study on iPinYou and Avazu to evaluate the effects of different components of FeSAIL on the performance. Similar conclusions can be drawn on Criteo and Media. Specifically, we consider the following settings.

TABLE IV: Average AUC & LogLoss over 10 test periods for four datasets. The results on different metrics are given by the average scores over all the incremental datasets. Note that "RI" indicates the relative improvement of FeSAIL over the corresponding baseline.

Datasets	Criteo			iPinYou			Avazu		
Methods	AUC	Logloss	RI (%)	AUC	Logloss	RI (%)	AUC	Logloss	RI (%)
IU	0.7329	0.5284	4.43	0.7528	0.4866	4.09	0.7210	0.4388	3.81
RS	0.7363	0.5198	3.41	0.7585	0.4791	2.96	0.7244	0.4346	3.11
RMFX	0.7415	0.5096	2.10	0.7621	0.4791	2.72	0.7306	0.4328	2.48
GAG	0.7459	0.5084	1.68	0.7674	0.4715	1.58	0.7352	0.4249	1.26
RMFX+SAR	0.7480	0.5081	1.50	0.7672	0.4728	1.73	0.7344	0.4251	1.34
GAG+SAR	0.7498	0.5012	0.71	0.7735	0.4701	1.04	0.7380	0.4215	0.68
EWC	0.7391	0.5124	2.53	0.7595	0.4783	2.81	0.7295	0.4252	1.69
ASMG	0.7522	0.5019	0.62	0.7713	0.4688	1.04	0.7414	0.4213	0.41
EWC+SAS	0.7479	0.5075	1.45	0.7664	0.4762	2.14	0.7343	0.4196	0.69
ASMG+SAS	0.7535	0.5001	0.35	0.7734	0.4666	0.68	0.7407	0.4200	0.30
FeSAIL	0.7553	0.4978	-	0.7788	0.4636	-	0.7420	0.4181	-

TABLE V: Average AUC and LogLoss results over 10 test periods on Media.

Methods	AUC	Logloss	RI (%)
IU	0.6316	0.2884	4.46
RS	0.6379	0.2842	3.26
RMFX	0.6346	0.2844	3.56
GAG	0.6401	0.2759	1.65
RMFX+SAR	0.6379	0.2804	2.61
GAG+SAR	0.6451	0.2729	0.71
EWC	0.6374	0.2797	2.53
ASMG	0.6452	0.2757	1.21
EWC+SAS	0.6398	0.2758	1.65
ASMG+SAS	0.6471	0.2727	0.52
FeSAIL	0.6503	0.2712	-

TABLE VI: Speed test.

Method/Runtime (min)	sampling	training	total
IU	0	2,519	2,519
FSS	190	2,519	2,709
reservior	143	920	1,063
reservior+SAR	143	1,014	1,157
reservior+SAS	401	920	1,321
FeSAIL	401	1,014	1,415

- **Incremental Update (IU)**[4] updates the model incrementally using only the new data.
- **Full stale sampling (FSS)** updates model with both new data and all historical samples containing stale features.

[4]For fair comparison, we supplement the latest historical samples to make the total samples size same to FSS

- **Reservior** updates the model using the new data and all the samples in reservior.
- **Rerservior+SAR** add the staleness aware embedding regularization on **Reservior**
- **Rerservior+SAS** restrict the reservior size of **Reservior** using staleness aware sampling.
- **FeSAIL** using both the SAR and SAS on **Reservior**.

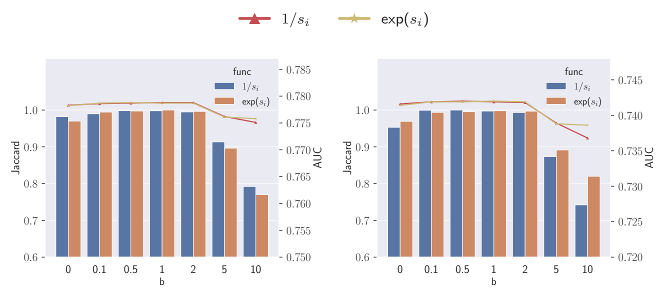
The results are presented in Figure 4. The line chart reflects the AUC difference between different methods. We can see that FeSAIL performs best among all the comparison methods, which shows the removal of any component from FeSAIL will hurt the final performance. SAR and SAS jointly contribute to improving the **Reservior** performance. The bar chart reflects the samples number in each time span. We can see that **FSS** samples size increase during incremental learning due to the growing size of historical samples. However, the samples size of **Reservior** and **FeSAIL** maintain relatively stable. Moreover, the samples size of **FeSAIL** is fixed and controllable, which is more friendly to real world incremental recommendation systems.

One concern of SAS is to control the time cost of sampling and model training. Table VI shows the average sampling time and the retraining time of each methods over different time spans on Avazu. The testing platform is given in Section III-E. We can see that the time cost for SAR is negligible compared to the total training time cost. SAS can remarkably reduce the samples size and accelerate the model retraining. In the meanwhile, thanks to neighbor-based SAS algorithm, the sampling time cost for SAS is bearable.

2) Effects of Field-wise SAS/SAR: We also perform an ablation study on Media to evaluate the effects of Field-wise SAS/SAR on handling the multi-categorial feature fields. Specifically, there are three choices for SAS module which are SAS (vanilla SAS), SASmean (set ϕ as mean function), and SASmax (set ϕ as max function). And there are also

TABLE VII: Ablation Study for Field-wise SAS/SAR on Media. The bold line indicates the group with best performance.

Methods		AUC	Logloss
SAS	SAR	0.6466	0.2763
	SARmean	0.6477	0.2741
	SARmax	0.6483	0.2735
SASmean	SAR	0.6487	0.2731
	SARmean	0.6588	0.2728
	SARmax	0.6495	0.2716
SASmax	SAR	0.6496	0.2721
	SARmean	0.6501	0.2717
	SARmax	**0.6503**	**0.2711**



(a) iPinYou (b) Avazu

Fig. 5: Parameter sensitivity *w.r.t* different inverse correlation function and bias on iPinYou and Avazu.

three choices for SAR module which are SAR (vanilla SAR), SARmean (set ψ as mean function), and SARmax (set ψ as max function). The average AUC and Logloss results on 10 test periods of all nine combinations are presented in Table VII. We can see that both SASmean and SASmax can improvement the overall performance, which shows it can alleviate the staleness feature imbalance problem for those datasets containing multi-categorial feature fields. The groups with max function performance slightly better than those with mean function, the reason for which might be that max function will not rescale the features with largest weight and these features are more necessary for model to overcome feature staleness problem. Similar conclusion can be obtained for SARmean and SARmax modules. We observe that the performance improvement acquired by field-wise SAS is larger than field-wise SAR. The reason might be that SAS module takes effect before model training and filters out a certain percentage of staleness feature, thus there are more staleness feature can be handled by SAS than SAR in each time span.

D. Parameter Sensitivity: RQ3

We investigate the sensitivity of choosing different inverse correlation function $func$ and bias b in Eq.(2). We choose iPinYou and Avazu two datasets and report the average AUC (line chart) and Jaccard Similarity (JS) of chosen samples set with the control group (the group with the highest AUC) over 10 time spans. The $func$ is chosen between inverse proportional function $1/s_i$ and negative exponential function $\exp(s_i)$. The b is chosen from $[0, 0.1, 0.5, 1, 2, 5, 10]$, where the larger b means th e less consideration for feature staleness. As shown in Figure 5, the performance on two datasets achieve the similar JS with two $func$ and b in range $[0.1, 0.5, 1, 2]$, which means the chosen samples set is insensitive to the $func$ and the b within a moderate range and productively achieve a competitive AUC towards the control group. The reason might be that only the relative weight sum order of samples will influence the choosing order in Algorithm 1 and choosing different $func$ and b with small values will not change the relative weight sum order.

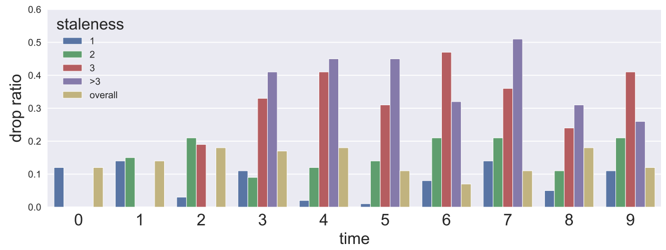
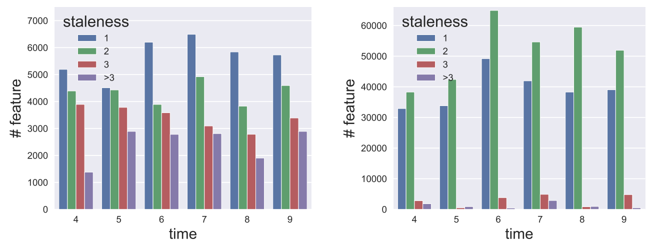


Fig. 6: The drop ratios of feature groups with different staleness on iPinYou.



(a) Avazu (b) Media

Fig. 7: The number of features with different staleness on Avazu and Media.

E. Case Study: RQ4

1) Effectiveness of SAS: We perform a case study on iPinYou to illustrate the effectiveness of SAS in selecting samples. As presented in Figure 6, we have two observations. First, the overall feature drop ratio after SAS is around 10%, which is consistent with the approximation ratio given by the Algorithm 1. Second, the features with larger staleness have higher drop ratio in most time spans except in the first several time spans which do not have features with staleness larger than its time span idx. It means SAS prefers to choose feature with smaller staleness, which is consistent with our motivation and mitigates the low frequency features influence on performance.

2) The Staleness Extent of Dataset: We also perform a case study on Avazu and Media to show that SAS can reflect the overall feature staleness extent of a dataset. As presented in Figure 7a, we can see that feature groups in Avazu are evenly distributed on different stalenesses, which means Avazu has a certain percentage of features reappearring after a long time and has high staleness extent. While in Figure 7b, the staleness of feature groups in Media are concentrate within small values, which means this dataset has relatively lower staleness extent.

V. Related Work

A. Continual Learning

Methods developed for continual learning can be generally divided into three categories: experience replay, regularization-based methods, and model expansion. Experience replay prevents forgetting by reusing past samples together with the new data to update the model [12], [13]. Then generative methods are developed to alleviate the burden of storing real data [15], [37]. Regularization-based methods preserve past knowledge by constraining the parameters update based on some measure of weight importance [36], [38]. Some works also employ knowledge distillation techniques to regularize the update direction [39], [40]. Model expansion supports continual learning of tasks by gradually incorporating subnetworks. Both expandable [41] and fixed-size network methods [42] are developed. Though these methods have shown convincing results in CV and NLP, it is still non-trivial to adapt them to incrementally update RS models, for the reason that they lack mechanism to explicitly optimize for the future performance. To overcome the specific forgetting problem in RSs incremental update, two groups of research have been developed in recent years. And they are closely related to some of the methods in continual learning [12], [20], [43].

B. Sample-based Incremental Methods for CTR prediction

This group of works relies on reusing historical samples to preserve the past memory, which is analogous to experience replay in continual learning. In most scenarios, a reservoir is maintained to keep a random sample of history [35]. After that, heuristic algorithms are designed to select samples from the reservoir for model updating, based on prioritizing recency [43], [44] or the extent of being forgotten [14], [45]. Sample-based approach can be seen as an intermediate between the incremental update and training on full historical data, which is aiming to find a balance between short-term interest and long-term memory. However, an apparent limitation of this approach is that the individual samples, though carefully selected with some measure of informativeness, are insufficient to represent the whole horizon of the overall distribution. On the contrary, the historical models learned from the historical data can be seen as a better summarization of the past knowledge. Hence, model-based approach which focuses on transferring knowledge between past and present models was later on developed to overcome these limitations.

C. Model-based Incremental Methods for CTR prediction

Model-based methods aim to overcome the forgetting problem when transferring knowledge between past and present models. Similar to the regularization-based methods in continual learning, these methods [18], [46] incorporates a knowledge distillation loss to regularize the model update, using previous incremental model serves as the teacher model. Other method [12], [47] employ the distillation techniques and specifically target at GNN-based and session-based RSs respectively. However, simply constraining the change in networks can not guarantee well that the knowledge which is useful for future prediction is extracted. Another distillation based method [48] proposed recently attempt to build a self-correction learning loop (dubbed ReLoop) for recommender systems. In particular, a new customized loss is employed to encourage every new model version to reduce prediction errors over the previous model version during training. In another group, a recently proposed method called Sequential Meta-Learning (SML) [20] achieves this by developing a mechanism to explicitly optimize the model parameters for the next period data. It devises a transfer module to jointly train the past and present models, which is adaptively evolving in a sequential manner for future serving. [21] argues that both SML and the distillation methods is that they can only consider transfer between models of two consecutive incremental periods, thus they try to modeling the long-term sequential patterns that may be highly valuable for generating a better model for future serving. However, both distillation methods and meta-learner based methods do not preserve the previously learned knowledge from the perspective of feature level, which may fail to perform well on the samples containing stale features.

VI. Conclusion

In this paper, we show the feature staleness problem existing in CTR prediction and propose a feature staleness aware incremental learning method for CTR prediction (FeSAIL) to mitigate this problem by adaptively replaying samples containing stale features. We firstly introduce a feature staleness aware sampling algorithm (SAS) to guarantee the sampling efficiency. We then introduce a feature staleness aware regularization mechanism (SAR) to restrict the low frequency feature updating. Furthermore, we enhance SAS and SAR to handle staleness imbalance for CTR prediction with multi-categorial feature fields. We conduct extensive experiments and demonstrate that FeSAIL can efficiently tackle the feature staleness problem. On average, it achieves 0.52% AUC improvement compared to various state-of-the-art methods on three widely-used public datasets and a private dataset with multi-categorial feature fields.

References

[1] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 1149–1154.

[2] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A Factorization-Machine based Neural Network for CTR Prediction," *arXiv*, 2017.

[3] W. Zhang, J. Qin, W. Guo, R. Tang, and X. He, "Deep learning for click-through rate estimation," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 8 2021, pp. 4695–4703.

[4] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer," *CIKM*, 2019.

[5] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep Interest Evolution Network for Click-Through Rate Prediction," *AAAI*, pp. 5941–5948, 2018.

[6] Y. Guo, F. Farooq, G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep Interest Network for Click-Through Rate Prediction," *AAAI*, pp. 1059–1068, 2018.

[7] C. Chen, H. Yin, J. Yao, and B. Cui, "TeRec: a temporal recommender system over tweet stream," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1254–1257, 2013.

[8] R. Barrett, R. Cummings, and E. Agichtein, "Streaming Recommender Systems," *Proceedings of the 26th International Conference on World Wide Web*, pp. 381–389, 2017.

[9] L. Zhang, H. Jiang, F. Wang, D. Feng, and Y. Xie, "T-Sample: A Dual Reservoir-based Sampling Method for Characterizing Large Graph Streams," *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, vol. 00, pp. 1674–1677, 2019.

[10] A. Teredesai, V. Kumar, and Y. Li, "Streaming Session-based Recommendation," *Proceedings of the 25th ACM International Conference on Knowledge Discovery & Data Mining*, pp. 1569–1577, 2019.

[11] F. Mi and B. Faltings, "Memory Augmented Neural Model for Incremental Session-based Recommendation," *arXiv*, 2020.

[12] F. Mi, X. Lin, and B. Faltings, "ADER: Adaptively Distilled Exemplar Replay Towards Continual Learning for Session-based Recommendation," *arXiv*, 2020.

[13] A. ROBINS, "Catastrophic Forgetting, Rehearsal and Pseudorehearsal," *Connection Science*, vol. 7, no. 2, pp. 123–146, 1995.

[14] J. Huang, Y. Chang, and X. a. Cheng, "GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation," *arXiv*, pp. 669–678, 2020.

[15] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual Learning with Deep Generative Replay," *arXiv*, 2017.

[16] A. Teredesai, V. Kumar, and Y. Li, "Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction," *Proceedings of the 25th ACM International Conference on Knowledge Discovery & Data Mining*, pp. 2671–2679, 2019.

[17] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019, clrev.

[18] Y. Wang, H. Guo, R. Tang, Z. Liu, and X. He, "A Practical Incremental Method to Train Deep CTR Models," *arXiv*, 2020.

[19] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, "Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation," *arXiv*, 2020.

[20] J. Huang, Y. Chang, and X. Cheng, "How to Retrain Recommender System? A Sequential Meta-Learning Method," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1479–1488, 2020.

[21] D. Peng, S. J. Pan, J. Zhang, and A. Zeng, "Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems," *Fifteenth ACM Conference on Recommender Systems*, pp. 411–421, 2021.

[22] B. Krishnapuram, M. Shah, and J. Mao, "Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 255–262, 2016.

[23] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & Cross Network for Ad Click Predictions," *Proceedings of the ADKDD'17*, pp. 1–7, 2017.

[24] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized Markov chains for next-basket recommendation," *WWW*, pp. 811–820, 2010.

[25] M. d'Aquin, S. Dietze, C. Hauff, E. Curry, P. C. Mauroux, X. Li, C. Wang, B. Tong, J. Tan, X. Zeng, and T. Zhuang, "Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction," *CIKM*, pp. 785–794, 2020.

[26] R. He and J. McAuley, "Fusing Similarity Models with Markov Chains for Sparse Sequential Recommendation," *arXiv*, 2016.

[27] W.-C. Kang and J. McAuley, "Self-Attentive Sequential Recommendation," *ICDM*, pp. 197–206, 2018.

[28] J. Caverlee, X. B. Hu, M. Lalmas, W. Wang, J. Li, Y. Wang, and J. McAuley, "Time Interval Aware Self-Attention for Sequential Recommendation," *WSDM*, pp. 322–330, 2020.

[29] C. Chekuri and A. Kumar, "Maximum coverage problem with group budget constraints and applications," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2004, pp. 72–83.

[30] S. Sen, W. Geyer, J. Freyne, P. Castells, P. Covington, J. Adams, and E. Sargin, "Deep Neural Networks for YouTube Recommendations," pp. 191–198, 2016.

[31] E.-P. Lim, M. Winslett, M. Sanderson, A. Fu, J. Sun, and S. Culpepper, "Neural Attentive Session-based Recommendation," *CIKM*, pp. 1419–1428, 2017.

[32] J. L. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," *ICLR*, pp. 785–794, 2015.

[33] W. Zhang, S. Yuan, J. Wang, and X. Shen, "Real-Time Bidding Benchmarking with iPinYou Dataset," *arXiv*, 7 2014.

[34] W. Zhu, D. Tao, X. Cheng, P. Cui, E. Rundensteiner, D. Carmel, Q. He, J. X. Yu, Z. Li, Z. Cui, S. Wu, X. Zhang, and L. Wang, "Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction," *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 539–548, 11 2019.

[35] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl, "Real-time top-n recommendation in social streams," *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*, pp. 59–66, 2012.

[36] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *arXiv*, 2016, ewc.

[37] P. Mozhgan, Z. Guoying, and S. Mohammad, "Looking Back on Learned Experiences For Class/task Incremental Learning," *ICLR*, 2022.

[38] F. Zenke, B. Poole, and S. Ganguli, "Continual Learning Through Synaptic Intelligence." *Proceedings of machine learning research*, vol. 70, pp. 3987–3995, 1 2017.

[39] C. Simon, P. Koniusz, R. Nock, and M. Harandi, "Adaptive Subspaces for Few-Shot Learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, pp. 4135–4144, 2020, icarl.

[40] J. Wen, Y. Cao, and R. Huang, "Few-Shot Self Reminder to Overcome Catastrophic Forgetting," *arXiv*, 12 2018.

[41] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *arXiv*, 6 2016.

[42] A. Mallya and S. Lazebnik, "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7765–7773, 6 2018.

[43] C. Chen, H. Yin, J. Yao, and B. Cui, "TeRec: a temporal recommender system over tweet stream," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1254–1257, 8 2013.

[44] Y. Zhao, S. Wang, Y. Wang, and H. Liu, "Stratified and time-aware sampling based adaptive ensemble learning for streaming recommendations," *Applied Intelligence*, vol. 51, no. 6, pp. 3121–3141, 6 2021.

[45] A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, G. Karypis, L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Q. V. Hung, "Streaming Session-based Recommendation," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1569–1577, 7 2019.

[46] K. Collins-Thompson, Q. Mei, B. Davison, Y. Liu, E. Yilmaz, W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen, "Streaming Ranking Based Recommender Systems," *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 525–534, 6 2018.

[47] M. d'Aquin, S. Dietze, C. Hauff, E. Curry, P. C. Mauroux, Y. Xu, Y. Zhang, W. Guo, H. Guo, R. Tang, and M. Coates, "GraphSAIL: Graph Structure Aware Incremental Learning forRecommender Systems," *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2861–2868, 10 2020.

[48] G. Cai, J. Zhu, Q. Dai, Z. Dong, X. He, R. Tang, and R. Zhang, "ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems," *arXiv*, 4 2022.