ReLoop2: Building Self-Adaptive Recommendation Models via Responsive Error Compensation Loop

Jieming Zhu[*]
Huawei Noah's Ark Lab
Shenzhen, China
jiemingzhu@ieee.org

Guohao Cai[*]
Huawei Noah's Ark Lab
Shenzhen, China
caiguohao1@huawei.com

Junjie Huang[†]
Shanghai Jiao Tong University
Shanghai, China
legend0018@sjtu.edu.cn

Zhenhua Dong
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

Ruiming Tang
Huawei Noah's Ark Lab
Shenzhen, China
tangruiming@huawei.com

Weinan Zhang
Shanghai Jiao Tong University
Shanghai, China
wnzhang@sjtu.edu.cn

ABSTRACT

Industrial recommender systems face the challenge of operating in non-stationary environments, where data distribution shifts arise from evolving user behaviors over time. To tackle this challenge, a common approach is to periodically re-train or incrementally update deployed deep models with newly observed data, resulting in a continual learning process. However, the conventional learning paradigm of neural networks relies on iterative gradient-based updates with a small learning rate, making it slow for large recommendation models to adapt. In this paper, we introduce ReLoop2, a self-correcting learning loop that facilitates fast model adaptation in online recommender systems through responsive error compensation. Inspired by the slow-fast complementary learning system observed in human brains, we propose an error memory module that directly stores error samples from incoming data streams. These stored samples are subsequently leveraged to compensate for model prediction errors during testing, particularly under distribution shifts. The error memory module is designed with fast access capabilities and undergoes continual refreshing with newly observed data samples during the model serving phase to support fast model adaptation. We evaluate the effectiveness of ReLoop2 on three open benchmark datasets as well as a real-world production dataset. The results demonstrate the potential of ReLoop2 in enhancing the responsiveness and adaptiveness of recommender systems operating in non-stationary environments.

CCS CONCEPTS

• **Information systems → Recommender systems**.

---

[*]Both authors contributed equally. Jieming Zhu is the corresponding author.
[†]Work done during internship at Huawei Noah's Ark Lab.

KEYWORDS

Recommender systems, continual learning, distribution shift, model adaptation, retrieval augmentation

1 INTRODUCTION

Nowadays, personalized recommendation has emerged as a prominent channel across a range of online applications, including e-commerce, news feeds, and music apps. It enables the delivery of tailored items to users based on their individual interests. The provision of high-quality recommendations not only enhances user engagement but also fuels revenue growth for platforms. To achieve accurate recommendations, deep learning-based models have gained widespread adoption in industry owing to their flexibility and ability to capture intricate user-item relationships. However, industrial recommender systems often operates in non-stationary environments, where data distribution shifts [36] occur as a result of evolving user behaviors over time. This can lead to the deterioration of well-trained recommendation models during online serving, and thus poses a challenge for models to quickly adapt under distribution shifts.

To address this challenge, previous research efforts have been made from two aspects: behavior sequence modeling and incremental model training. The first line of research aims to capture dynamic patterns at the feature level by modeling sequential user behavior sequences. Notable progress has been made in this area, particularly in click-through rate (CTR) prediction tasks [68], with models incorporating attention, GRU, and transformer architectures, such as DIN [67], DIEN [66], and BST [5]. These models formulate CTR prediction as a few-shot learning task [62], where given k historical behaviors from a user (i.e., k-shot samples), the goal is to predict whether the user will click on the next item. Few-shot learning enables rapid adaptation to new users with only a few observed samples [32]. However, these studies do not explicitly address the test-time distribution shift problem. In parallel, another

line of research focuses on incremental model training. While regular re-training of models (e.g., daily) is straightforward, it becomes time-consuming due to the large volume of training data in practice (e.g., up to billions of samples in Google Play's app recommendation [7]). As a result, incremental model training [26, 47, 61] has gained popularity in industrial recommender systems. This approach retains previous model parameters for initialization [26] or knowledge transfer [47] while continually updating the model using newly observed data samples, often at a minute-level granularity. Incremental training brings training efficiency and enhances model freshness. However, learning the parameters of neural networks relies on iterative gradient-based updates to gradually incorporate supervision information into model weights with a small learning rate. This makes it challenging for large parametric recommendation models to achieve fast adaptation to distribution shifts. This challenge, often referred to as the stability-plasticity dilemma [37] in incremental learning, stems from the need to balance the stability of existing knowledge with the plasticity required to incorporate new information efficiently.

In comparison, humans possess an extraordinary capability for fast incremental learning in dynamic environments [1]. This remarkable learning ability can be attributed to the presence of two complementary learning systems (CLS) in the human brain: the *hippocampus* and the *neocortex* [30, 44]. The hippocampus is responsible for rapid learning of recent specific experiences and exhibits short-term adaptation. On the other hand, the neocortex functions as a slow learning system, gradually acquiring structured knowledge about the environment over time. The combination of these slow and fast CLS learning mechanisms empowers humans to learn quickly and remember information in the long term. This inspires us to design an adaptive recommendation framework that incorporates both fast and slow learning modules to build self-adaptive recommendation models and make accurate recommendations in dynamic environments.

Our approach consists of two key components: a base model serving as a slow learning module that updates through gradient back-propagation, and a fast learning module equipped with a non-parametric error memory. Unlike traditional gradient-based training, the fast learning module is training-free and thus enables rapid adaptation to new data distributions. To be specific, consider the learning process of students, who often organize and review their past incorrect questions to reflect on their errors and improve their performance in subsequent exams. Inspired by this, we propose to store recent error samples from the incoming data stream in the error memory. These error samples reflect situations where the model's performance degrades, particularly during distribution shifts. In response, we estimate the errors between the base model's predictions and the ground-truth labels using the error memory. These error estimations are then utilized to compensate for the model's degradation caused by distribution shifts. During the model serving phase, newly observed data is continuously written to the error memory, creating a self-correcting learning loop that facilitates fast model adaptation. We refer to this approach as ReLoop2, which extends the original ReLoop learning framework [3] to test-time adaptation.

In building a self-correcting learning loop for online recommendation, we encounter two primary challenges. The first challenge

pertains to estimating the errors for output compensation without relying on back-propagation. To address this, we propose a non-parametric method that leverages the error memory to retrieve similar samples, enabling us to approximate the errors effectively. The second challenge lies in the time- and space-efficient design of the error memory. Given the substantial volume and high velocity of data streams, we introduce a locality-sensitive hashing (LSH)-based sketching approach. This approach ensures efficient $O(1)$-time memory reading and writing operations while maintaining a constant memory footprint.

The ReLoop2 framework establishes a continual model adaptation process by continuously refreshing the error memory with newly observed error samples after model deployment. It has the potential to significantly enhance model performance when faced with data distribution shifts. Importantly, the framework is orthogonal to existing incremental learning techniques and compatible with diverse models used in recommendation systems. We empirically validate the effectiveness of ReLoop2 on three open benchmark datasets and a proprietary production dataset, showcasing substantial performance improvements over existing models and incremental training techniques. We hope that our work could inspire further research attention to address the challenge of train-test distribution shift in online recommender systems. In summary, this paper makes three main contributions:

- We identify the challenging problem of fast model adaptation for online recommendation, and propose a slow-fast learning paradigm inspired by the complementary learning systems observed in human brains.
- We introduce a time- and space-efficient non-parametric error memory and leverage it to build a responsive error compensation loop for fast model adaptation.
- We conduct extensive experimental evaluations on both open benchmark datasets and real-world production datasets to demonstrate the effectiveness of our ReLoop2 framework.

2 BACKGROUND

In this section, we provide an overview of the CTR prediction task. We then describe our motivation for fast model adaptation in real-world scenarios. Additionally, we review the locality-sensitive hashing (LSH) technique used in our work.

2.1 CTR Prediction

Typically, input samples consist of two main types of features: categorical features, and numerical features. In our approach, we utilize embedding techniques to map these features into a lower-dimensional embedding space. Specifically, for numerical features, we first discretize them into categorical bucket features. Then, same with categorical features, we apply one-hot/multi-hot encoding and embedding table lookups to obtain embedding vectors. Let $x = \{x_1, x_2, ..., x_n\}$ denotes a data instance with n feature fields. Then we could get its feature embeddings as $e = \{e_1, e_2, ..., e_n\}$, which serve as the input to a deep neural network.

Given the set of feature embeddings, various CTR prediction models have been proposed to model feature interactions (e.g., DeepFM [12], DCN [58], DCN-V2 [59]) and sequential user interests (e.g., DIN [67], DIEN [66], and BST [5]). At last, the CTR model

outputs the predicted click probability $\hat{y} \in [0,1]$ using the sigmoid activation $\sigma(\cdot)$ on the logit value. Formally, we denote a CTR prediction model as follows:

$$\hat{y} = \sigma\big(\phi(e)\big) \tag{1}$$

where $\phi(\cdot)$ is a multi-layer deep neural network. For example, ϕ_{DeepFM}, ϕ_{DCN}, and ϕ_{DIN} are commonly used model architectures [12, 58, 67].

We denote $y \in \{0,1\}$ as a true label to indicate whether a user has clicked a recommended item. The binary cross-entropy loss function is usually used for CTR prediction:

$$L = -\frac{1}{N} \sum \big(y log\hat{y} + (1-y)log(1-\hat{y})\big) \tag{2}$$

where N is the number of training instances. Readers may refer to our BARS benchmark [68] for more training details.

2.2 Fast Model Adaptation

In this section, we analyze the motivation behind the need for fast model adaptation to address the problem of distribution shift. In Figure 1, we present our observations regarding the dynamic data distribution from various perspectives, including data variance, feature dynamics, overall CTR, and category-specific CTR over time. To illustrate this, we split the test dataset of MicroVideo (detailed in Section 4) into ten chronological time slots, simulating an online advertising scenario. Specifically, Figure 1(a) depicts the data variance (from embedding e) for each time slot, revealing the spread of data instances relative to their average. A higher value indicates a greater deviation from the average. Figure 1(b) demonstrates the changes in the number of users and items over time. Both (a) and (b) highlight the covariate shift in feature x.
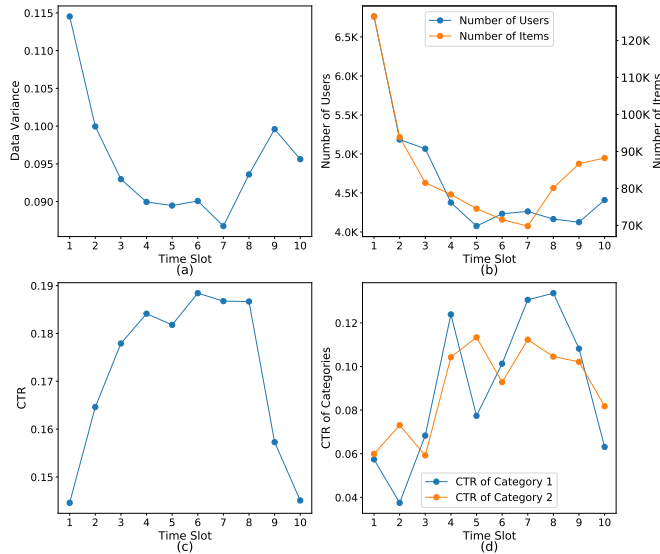
Figure 1: Observations of data distribution shifts on the MicroVideo dataset. (a) Variance of data samples over each time slot. (b) Number of users and items involved over each time slot. (c) The averaged CTR over each time slot. (d) The averaged CTR of two unique categories over each time slot.

Furthermore, Figure 1(c) showcases the dynamic nature of CTR over time, while (d) exhibits the dynamic CTR based on different categories. These figures reveal the label shift in y and the concept drift between x and y, respectively. Collectively, these visualizations demonstrate a significant level of data change occurring over time. As a result, there is a pressing demand for fast model adaptation to swiftly adjust to the dynamic patterns present in the data.

2.3 Locality Sensitive Hashing

This section provides a brief review of the classical Locality Sensitive Hashing (LSH) algorithm [9, 21], which is a widely adopted sublinear-time algorithm for approximate nearest neighbor search. In LSH, a hash function $h(x) \mapsto \mathbb{Z}$ a mapping that assigns an input x to an integer in the range $[0, R-1]$. LSH encompasses a family of such hash functions with a key property: similar points have a high probability of being mapped to the same hash value [9]. More formally, a LSH family is defined as follows [9].

DEFINITION 1. **LSH Family**. *A family $\mathcal{H}$ is called (S_0, cS_0, p_1, p_2)-sensitive with respect to a similarity measure $sim(\cdot, \cdot)$ if for any two points $x, y \in \mathbb{R}^d$ and h chosen uniformly from $\mathcal{H}$ the following properties hold:*

- *if $sim(x,y) \geq S_0$ then $p(x,y) \geq p_1$*
- *if $sim(x,y) \leq cS_0$ then $p(x,y) \leq p_2$*

Typically, $p_1 > p_2$ and $c < 1$ is required for approximate nearest neighbor search. We use the notation $p(x,y)$ to denote the collision probability $Pr\big(h(x) = h(y)\big)$ between x and y, where their hash values are equal. One sufficient condition for being a LSH family is that the collision probability $p(x,y)$ is a monotonically increasing function of similarity between the two data points, i.e.,

$$p(x,y) = f\big(sim(x,y)\big) \tag{3}$$

where $f(\cdot)$ is required to be a monotonically increasing function. In other words, similar data points are more likely to collide with each other under LSH mapping.

Among the widely known LSH families, SimHash [18] is a popular LSH that applies the technique of Signed Random Projections (SRP) [4, 10, 18] for the cosine similarity measure. Given a vector x, SRP utilizes a random w vector with each component generated from i.i.d. normal, i.e., $w_i \sim N(0,1)$, and only stores the sign of the projection. Hence, SimHash is given by

$$h_w(x) = sign(w^T x) . \tag{4}$$

Particularly, we could take $[h_w(x)]_+ = max\big(0, h_w(x)\big)$ using the ReLU function to re-map it to $\{0,1\}$. It has been shown in the seminal work [10] that the collision probability under SRP satisfies the following equation:

$$p(x,y) = 1 - \frac{1}{\pi}cos^{-1}\big(\frac{x^T y}{\|x\|_2 \|y\|_2}\big) \tag{5}$$

where $p(x,y)$ is monotonic to the cosine similarity $\frac{x^T y}{\|x\|_2 \|y\|_2}$.

It is important to note that each hash function $h_w(x)$ generates a single bit using SRP, resulting in two possible hash values $\{0,1\}$. By independently sampling L hash functions with different w vectors, we can generate new hash values in the range $[0, 2^L - 1]$ by combining the outcomes of the L independent SRP bits. The collision probability is equal to $p(x,y)^L$, the power of L of Equation 5.
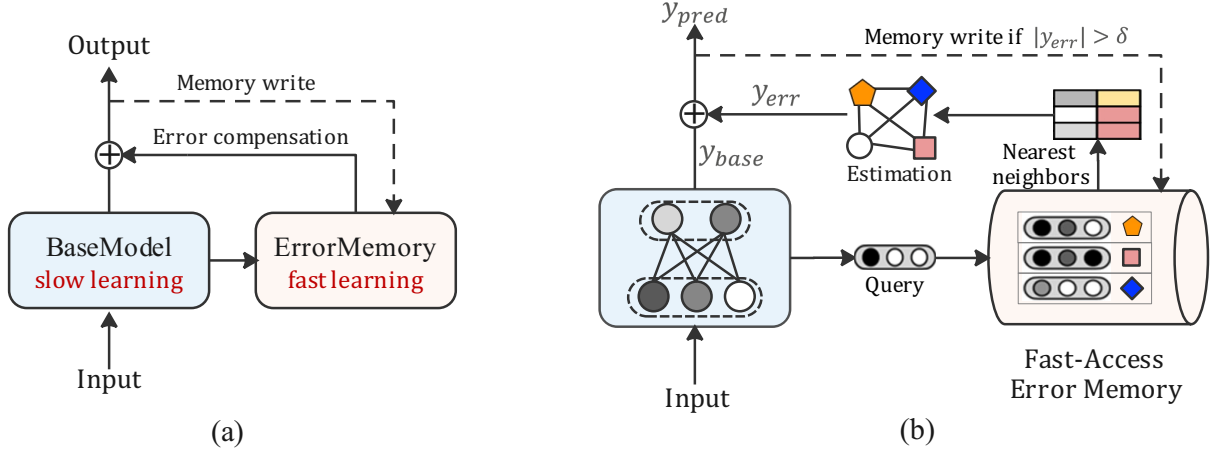
Figure 2: (a) An overview of our slow-fast learning framework, which comprises a slow-learning base model and a non-parametric error memory module for fast adaptation; (b) The ReLoop2 diagram that builds a self-correcting learning loop with error compensation.

3 APPROACH

In this section, we present our ReLoop2 apporach that enables fast model adaptation through a self-correcting learning loop with responsive error compensation.

3.1 Overview

Deep learning-based recommendation models, such as CTR prediction models discussed in Section 2.1, are typically optimized using back-propagation algorithms within the empirical risk minimization (ERM) framework. These models assume a stationary data distribution (i.e., the training and testing data are drawn from the same distribution) and require a small learning rate to gradually incorporate information into model weights. However, in real-world online recommendation scenarios, the rapid emergence of new users and items, along with potential changes in user behavior over time, result in the distribution shift challenge. Consequently, a well-trained model may gradually degrade after deployment. To address this challenge, we propose the ReLoop2 framework for fast model adaptation, as depicted in Figure 2.

Our framework employs a slow-fast learning paradigm, where the base model undergoes slow gradient updates, while an episodic memory module, free from back-propagation, is introduced to facilitate rapid acquisition of new knowledge. The base model is a standard parametric neural network that learns common knowledge through gradual gradient updates. In contrast, the memory module is a non-parametric component that stores recently observed samples and enables fast learning and adaptation from these samples. This slow-fast learning paradigm aligns with the theory of complementary learning systems (CLS) in human brains [30, 44].

Specifically, we refer to the memory module as the error memory, which stores the recent error samples produced by the base model on the incoming data stream. These error samples provide insights into cases where the model makes incorrect predictions, particularly in the presence of distribution shifts. By directly capturing and remembering these error samples, we can estimate errors in a

non-parametric manner and subsequently correct the model output through error compensation. This establishes a continual fast adaptation process for the model within the evolving dynamics of the non-stationary data distribution. New error samples observed during model deployment are continuously written back to the error memory, enabling the tracking of changing dynamics in the online data.

3.2 Error Compensation Loop

Figure 2(b) depicts our error compensation loop for fast model adaptation, comprising three key components: the base model ϕ, the fast-access error memory M, and the error estimation module $\mathcal{E}$. Our learning framework is generic and compatible with various base models used for CTR prediction. We formulate the base model as follows:

$$y_{base} = \phi(e) \tag{6}$$

where e represents the feature embeddings of a data instance. $y_{base} \in [0, 1]$ denotes the predicted click probability from the base model. We provide a brief overview of feature embedding and CTR modeling approaches in Section 2.1. It is worth noting that the model function ϕ can be implemented using any existing CTR prediction model, such as DeepFM [12], DCN [58], and DIN [67]. The base model approximates the ground truth label y by minimizing the error between y and y_{base} within an empirical risk minimization framework:

$$\min \ \epsilon^2 \,, \quad \text{where} \ \epsilon = y - y_{base} \tag{7}$$

Ideally, under the assumption of independent and identically distributed (i.i.d.) data, the error ϵ should be a small random variable close to zero after model training. However, the base model degrades when confronted with distribution shifts, resulting in an enlarged error ϵ during model serving.

To address this issue, we propose a proactive approach to estimate the model prediction error and correct the model output accordingly. However, directly obtaining ϵ from Equation 7 is not

feasible due to the unknown label y during prediction. Therefore, we propose to estimate it using recently observed samples stored in the error memory module M. Formally, we perform error estimation with the following formula:

$$y_{err} = \mathcal{E}(h_q, M) \tag{8}$$

where h_q denotes the hidden representation of input sample q, which can be chosen from any hidden layer (e.g., the last hidden layer) of the base model ϕ. With the estimated error y_{err}, we can make compensation for the model output to correct its prediction:

$$y_{pred} = y_{base} + \lambda \cdot y_{err} \tag{9}$$

where y_{pred} denotes the final output with model adaptation. The compensation weight λ adjusts the proportion of error compensation. It is important to note that the value of y_{pred} may exceed the range [0, 1] after error compensation. In such cases, we clamp the value within the range.

In the following sections, we will describe our designs for the error estimation module $\mathcal{E}$ and the error memory module M.

3.2.1 Error Estimation Module.

Given the error memory that retains recently observed data samples, our goal is to estimate the prediction error based on similar samples to a new input q. Formally, we aim to retrieve a set of top-k similar samples from the memory, as described below:

$$\mathcal{K} = \big\{(s_i, \ y_i, \ y_{base\_i}) \mid i \in M\big\} \tag{10}$$

where $s_i = sim(h_q, h_i)$ denotes the similarity between the hidden vectors of the query sample x and memory instance i. Additionally, y_i and y_{base_i} represent the ground-truth label and the prediction value of the base model, respectively. The derivation of similar samples $\mathcal{K}$ is provided in Section 3.2.2.

Inspired by the attention mechanism employed in content-addressing memory networks [63], we can estimate the attention-weighted ground truth $\bar{y}$ and prediction value $\bar{y}_{base}$ as follows.

$$\bar{y} = \sum_{i \in \mathcal{K}} a_i \cdot y_i, \quad \bar{y}_{base} = \sum_{i \in \mathcal{K}} a_i \cdot y_{base\_i} \tag{11}$$

The attention weight a_i is computed using the following equation:

$$a_i = \frac{exp(s_i/\tau)}{\sum_{i \in \mathcal{K}} exp(s_i/\tau)} \tag{12}$$

Here, τ is a temperature parameter that adjusts the smoothness of the softmax. The value of τ can be learned jointly with the base model or manually tuned as a hyper-parameter (e.g., 0.1).

Next, we estimate the prediction error as a weighted combination of two error measures:

$$\begin{aligned} y_{err} &= \gamma \cdot (\bar{y} - y_{base}) + (1 - \gamma) \cdot (\bar{y}_{base} - y_{base}) \\ &= \gamma \cdot \bar{y} + (1 - \gamma) \cdot \bar{y}_{base} - y_{base} \end{aligned} \tag{13}$$

where γ is a weight that balances the two error measures. Notably, when $\gamma = 0$, the error is estimated from the model predictions on similar samples. When $\gamma = 1$, the error is computed from the ground truth labels of similar samples. In the latter case, we substitute y_{err} into Equation 9 and can obtain the corrected model prediction with error compensation as follows:

$$\begin{aligned} y_{pred} &= y_{base} + \lambda \cdot (\bar{y} - y_{base}) \\ &= (1 - \lambda) \cdot y_{base} + \lambda \cdot \bar{y} \end{aligned} \tag{14}$$

This can be seen as a weighted ensemble of the base model's output y_{base} and the estimation $\bar{y}$ from k-nearest neighbors (KNN). For simplicity, we use $\gamma = 1$ in our experiments.

3.2.2 Fast-Access Error Memory.

In this section, we focus on the key component of our framework, the error memory M. In online recommendation systems, data is acquired sequentially over time, and the model generates click predictions based on the received samples. The true labels are received when users interact with the recommended items. During this process, we can easily obtain the hidden representation h_i from a specific hidden layer of the base model (same with h_q), the base model output y_{base_i}, and the true label y_i. To enable fast adaptation to distribution shifts, we utilize the memory to store these recently observed samples. Ideally, our memory consists of a set of key-value pairs formulated as follows:

$$M = \big\{(h_i, \ y_i, \ y_{base\_i})\big\} \tag{15}$$

where $h_i \in \mathbb{R}^d$ represents the d-dimensional key vector of memory slot i, while y_i and y_{base_i} serve as the memory values. We define a memory reader function R that retrieves a set of similar samples $\mathcal{K}$ from the memory given h_q as a query:

$$\mathcal{K} = R(h_q, \ M) \tag{16}$$

However, designing the memory poses two main challenges in real-world recommender systems due to the large volume of click data:

- **Fast Access**. For real-time online CTR prediction, model inference must meet stringent latency requirements. Therefore, it is crucial to enable fast access to the error memory. However, retaining a large number of recently observed data samples for adaptation makes the memory size too large to utilize traditional attention-based content addressing mechanisms in memory networks [63], which needs to read all memory slots for each query.
- **Memory Size**. The memory module requires a substantial memory size to store an adequate number of data samples. In our production system, the number of samples can easily reach millions within a 10-minute timeframe. Storing such a large number of data samples consumes significant computing resources (e.g., RAM) for model serving. Therefore, minimizing memory resource consumption for the error memory is highly desirable.

To address these challenges, we explore two potential solutions: approximate nearest neighbor (ANN) search and LSH-based sketching. ANN search techniques are widely used in industry to efficiently retrieve top-k nearest vectors in sub-linear time. These techniques have been successful in various retrieval-augmented machine learning tasks (e.g., language modeling [29], machine translation [27]). Additionally, they are supported by mature tools and libraries, including Faiss [23], ScaNN [13] and Milvus [56]. However, constructing popular ANN indices like HNSW [42] and IVFPQ in Faiss [23] involves time-consuming steps (e.g., k-means) and requires substantial memory (gigabytes of RAM). To reduce memory consumption, we propose filtering the data samples based on the model's errors. Specifically, we only store samples with relatively large errors (greater than a threshold σ) since they indicate significant degradation in the base model. Despite applying error filtering

and random down-sampling, storing these raw data samples still imposes a considerable burden on memory. Therefore, for online recommendation tasks with limited computing resources, ANN search may not be the optimal choice.

Ideally, we aim to design a lightweight and fast-access memory that avoids directly storing massive data points in RAM, eliminates the need for iterative and non-streaming processes like k-means, and avoids constructing complex index structures such as graphs, which are either time-consuming or difficult to parallelize. To achieve these objectives, we propose an alternative design of the error memory by employing the LSH-based data sketching algorithm on the streaming data [8]. LSH, as introduced in Section 2.3, enables efficient bucketing of each data point in constant time using fixed hash functions. Data sketching supports the construction of a compact sketch that summarizes the streaming data. The sketching algorithm compresses a set of high-dimensional vectors into a small array of integer counters, which is sufficient for estimating the similarity s_i of similar samples in Equation 16.

Formally, we define the memory as a sketch consisting of K repeated arrays, denoted as M_k for $k \in [0, K-1]$. Each array M_k is indexed by L independent hash functions $H_L(x) = \{h_w(x) \mid w\}$, where $h_w(x) \mapsto [0, 1]$ is a singed random projection described in Equation 4. Consequently, an input x can be hashed into an index in 2^L buckets: $H_L(x) \mapsto [0, 2^L - 1]$. For example, setting $L = 16$ can result in approximately 65,536 buckets. While the sketch is originally designed for kernel density estimation with integer counters [8], in our design, we store a tuple of summation values at bucket b in the array M_k, denoted as $M_k[b] = (sum\_x[b], sum\_y[b], sum\_y_{base}[b])$. To ensure more stable estimations, the same process is repeated K times using K different sets of hash functions $\{H_L^k(x) \mid k \in [0, K-1]\}$. In summary, the memory can be viewed as a concatenated array of size $2^L \times K \times 3$. More specifically, we formulate the memory writing and reading processes as follows:

Memory Writing. For each observed sample i from the data stream, we obtain (h_i, y_i, y_{base_i}). Instead of directly storing the raw samples in the memory following Equation 15, we apply each set of hash functions H_L^k to map the key vector h_i to the corresponding bucket b and update the sketch array $M_k[b]$ as follows:

$$\begin{aligned} sum\_x[b] &+= 1 \\ sum\_y[b] &+= y_i \\ sum\_y_{base}[b] &+= y_{base\_i} \end{aligned} \tag{17}$$

Note that the values in $M_k[b]$ are initially set to zero and can be reset regularly or when the base model has been updated to refresh the memory. The updates on all K memory arrays can be performed in parallel.

Memory Reading. Given a query sample vector h_q, we can apply the same set of hash functions to map the query to bucket b. We then obtain the summation values from each sketch array $M_k[b]$ and compute the readout values via averaging them over all buckets as follows:

$$\begin{aligned} s_i &= sum\_x[b] \ / \ \sum_b sum\_x[b] \\ y_i &= sum\_y[b] \ / \ \sum_b sum\_x[b] \\ y_{base\_i} &= sum\_y_{base}[b] \ / \ \sum_b sum\_x[b] \end{aligned} \tag{18}$$

After parallel reading from K sketch arrays, we obtain the K readout results of similar samples $\mathcal{K} = \{(s_i, y_i, y_{base\_i}) \mid i \in [0, K-1]\}$, which can then be used in Equation 10 for error estimation.

Compared to traditional memory that stores raw samples, our LSH-based sketch memory offers several advantages. It enables fast construction time (O(1) writing time per sample), has a low memory requirement (constant memory size of $2^L \times K \times 3$), and eliminates the need for query-time distance computations (O(1) reading time per query). It is worth noting that our sketch memory is not only practical to implement but also enjoys strong theoretical guarantees [8].

In this way, the error memory module helps estimate the potential error of the base model based on observed similar data samples, contributing to an error compensation loop that continuously and adaptively corrects the model output. This results in a slow-fast joint learning framework for fast model adaptation.

4 EXPERIMENTS

In this section, we present extensive experimental results conducted on three open benchmark datasets and one real-world production dataset to validate the effectiveness of ReLoop2. Our experiments aim to answer the following three research questions:

- **RQ1**: How does the integration of ReLoop2 with state-of-the-art models contribute to the improvement of model performance?
- **RQ2**: How does ReLoop2 compare to incremental training in terms of performance?
- **RQ3**: How do different hyperparameters affect model performance?

4.1 Experimental Setup

Datasets. We conduct experiments on three open benchmark datasets, and a large-scale production dataset.

- **AmazonElectronics** is a subset of the Amazon dataset [16], a widely used benchmark dataset for recommendation. We follow the DIN work [67] to preprocess the dataset. Specifically, the AmazonElectronics contains 1,689,188 samples, 192,403 users, 63,001 goods and 801 categories. Features include goods_id, category_id, and their corresponding user-reviewed sequences: goods_id_list, category_id_list.
- **MicroVideo** is an open dataset for short video recommendation, which has been released by [6]. We follow the same preprocessing steps. It contains 12,737,617 interactions that 10,986 users have made on 1,704,880 micro-videos. The labels include click or non-click, while the features include user_id, item_id, category, and the extracted image embedding vectors of cover images of micro-videos.
- **KuaiVideo** is another open dataset for short video recommendation. We follow the work [33] to obtain the preprocessed dataset. Specifically, we randomly selected 10,000 users and their 3,239,534 interacted micro-videos. It contains several interaction data between users and videos, such as user_id, photo_id, duration_time, click, like, and so on. In addition, 2048-dimensional video embeddings are provided as content features.
- **Production** is a production dataset from Huawei's news feed recommendation. It has a total of 500 million records sampled from 7 days user logs and each record has more than 100 fields of

Table 1: Performance comparison against the state-of-the-art models. BASE+ReLoop2 represents the setting that ReLoop2 is integrated to the best performing base model on each dataset.

Model	AmazonElectronics				MicroVideo				KuaiVideo			
	gAUC(%)	RelImp	AUC(%)	RelImp	gAUC(%)	RelImp	AUC(%)	RelImp	gAUC(%)	RelImp	AUC(%)	RelImp
FM	84.94	0%	84.85	0%	67.24	0%	71.86	0%	65.99	0%	74.18	0%
FmFM	85.29	0.4%	85.47	0.7%	67.37	0.2%	72.15	0.4%	65.52	-0.7%	73.89	-0.4%
DeepFM	87.89	3.5%	88.16	3.9%	68.52	1.9%	73.37	2.1%	66.65	1.0%	74.52	0.5%
DCN	87.78	3.3%	88.01	3.7%	68.55	1.9%	73.42	2.2%	66.58	0.9%	74.61	0.6%
xDeepFM	87.90	3.5%	88.13	3.9%	68.89	2.5%	73.62	2.4%	66.96	1.5%	74.71	0.7%
AutoInt+	87.87	3.4%	88.04	3.8%	68.46	1.8%	73.38	2.1%	66.67	1.0%	74.69	0.7%
DCNv2	87.90	3.5%	88.12	3.9%	68.59	2.0%	73.44	2.2%	66.75	1.2%	74.70	0.7%
AOANet	87.91	3.5%	88.12	3.9%	68.58	2.0%	73.46	2.2%	66.79	1.2%	74.70	0.7%
DIN	88.35	4.0%	88.60	4.4%	68.83	2.4%	73.60	2.4%	66.96	1.5%	74.95	1.0%
DIEN	88.56	4.3%	88.88	4.7%	68.67	2.1%	73.21	1.9%	67.11	1.7%	75.04	1.2%
BST	88.41	4.1%	88.64	4.5%	68.54	1.9%	73.42	2.2%	66.90	1.4%	74.84	0.9%
BASE+ReLoop2	**89.33**	**5.2%**	**89.62**	**5.6%**	**69.53**	**3.4%**	**74.11**	**3.1%**	**67.18**	**1.8%**	**75.13**	**1.3%**

Table 2: Evaluation of ReLoop2 across different base models.

Model	AmazonElectronics				MicroVideo			
	Base		+ReLoop2		Base		+ReLoop2	
	gAUC	AUC	gAUC	AUC	gAUC	AUC	gAUC	AUC
xDeepFM	87.90	88.13	**88.73**	**88.96**	68.89	73.62	**69.53**	**74.11**
DCNv2	87.90	88.12	**88.81**	**88.99**	68.59	73.44	**69.71**	**74.25**
AOANet	87.91	88.12	**88.75**	**88.94**	68.58	73.46	**69.11**	**73.86**
DIN	88.35	88.60	**89.10**	**89.34**	68.83	73.60	**69.22**	**73.86**
DIEN	88.56	88.88	**89.33**	**89.62**	68.67	73.21	**69.15**	**73.60**

features, such as doc_id, category_id, short-term interest topic_id, and anonymous data masking user_id. We use the latest 2-hour samples as testing data, and split it into 12 consecutive parts in chronological order.

Base models. We compare our model with the following mainstream base models for CTR prediction.
- Shallow models: FM [53], FmFM [55].
- Feature interaction models: DeepFM [12], xDeepFM [35], DCN [58], AutoInt+ [54], DCNv2 [59], AOANet [54].
- Behavior sequence models: DIN [67], DIEN [66], BST [5].

Metrics. We adopt the most popular ranking metrics, AUC [7] and gAUC [67] (i.e., user-grouped AUC), to evaluate the model performance. In addition, we report the relative improvements (RelImp) over the classic factorization machine (FM) model.

We emphasize that the preprocessed datasets and evaluation settings for all the baseline models we studied are available on the BARS benchmark website: https://openbenchmark.github.io/BARS.

4.2 Performance Evaluation with SOTA Models
We evaluate the ReLoop2 module on existing models, including many state-of-the-art (SOTA) methods. The performances are shown

in Table 1. Through the analysis of experiment results, we get some conclusions as follows: deep-learning-based methods get higher accuracy than the low-rank-based methods, thus revealing the powerful feature interaction ability of neural networks. In addition, xDeepFM obtains the second-best results on the MicroVideo dataset, indicating that a well-designed structure could fully use the advantages of the factorization machine component. What's more, DIEN method obtains the second-best results on AmazonElectronics and KuaiVideo datasets, which benefits from the evolution of user interests and exploitation of the sequential features. In addition, we can see that our BASE+ReLoop2 outperforms all the other baseline methods since the error memory module is applied to the baseline method to augment the base encoder, and the error compensation helps to adapt to data distribution shift rapidly. Specifically, we choose DIEN as the base model for AmazonElectronics and KuaiVideo, and DCNv2 for MicroVideo because of their relatively better performance. It is worth noting that our ReLoop2 framework is model agnostic to all the existing models, which is shown in Table 2. After applying ReLoop2 to the five state-of-the-art models respectively, we can obtain the new SOTA.

Evaluation on production dataset. The comparison of our model with the baseline on the product dataset is shown in Figure 3. The baseline is an incremental learning method, which serves as the base encoder, so the results of the first part of the test set are exactly the same. From the second part, we utilize the previous parts as fast access error memory to learn the error compensation rapidly, and the performance demonstrates the efficiency of our ReLoop2 apporach.

4.3 Evaluation between ReLoop2 and Incremental Training
As mentioned earlier, incremental model training has been a common choice in real-world production systems, so in Figure 4, we compare our fast model adaptation with incremental training based on DCNv2 backbone on MicroVideo. The horizontal axis of Figure 4
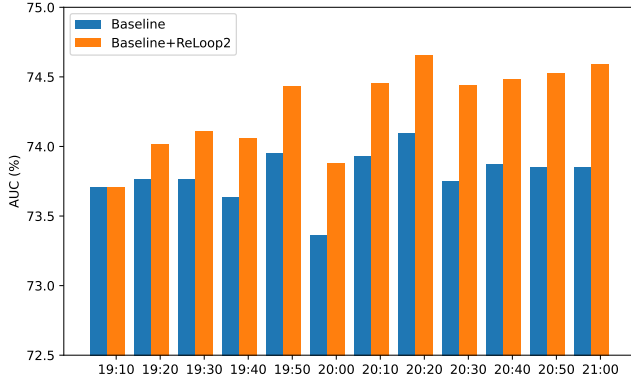
Figure 3: Evaluation of ReLoop2 on our production dataset.

(a) gAUC on MicroVideo

(b) AUC on MicroVideo

Figure 4: Comparison between ReLoop2 and incremental training on MicroVideo.

(a) Effect of Top-K on AmazonElectronics

(b) Effect of Top-K on MicroVideo

Figure 5: Effect of K on AmazonElectronics and MicroVideo.

(a) Effect of λ on AmazonElectronics

(b) Effect of λ on MicroVideo

Figure 6: Effect of compensation weight λ on AmazonElectronics and MicroVideo.

is time slot, as we split the test dataset of MicroVideo into ten time slots in chronological order evenly to simulate online advertising task. Four methods are compared in Figure 4.

- **DCNv2** is the baseline model in this experiment.
- **DCNv2+IncCTR** applies the incremental training method, IncCTR [61], on top of DCNv2. Specifically, after model training on the training data and model evaluation on the first part of the test data, we continually train the model using the first part of the test data and then evaluate it on the second part. The process goes on like this on ten test parts. Note that we only pass the test data once for IncCTR as suggested in the paper.
- **DCNv2+ReLoop2** applies fast model adaptation (FMA) to DCNv2.
- **DCNv2+IncCTR+ReLoop2** applies both incremental training and fast model adaptation (FMA) to DCNv2. Note that our ReLoop2 framework is orthogonal to the incremental training technique since ReLoop2 do not need extra training.

In Figure 4, ReLoop2 outperforms IncCTR most of the time on both gAUC and AUC, except for the last two time slots, where AUC of IncCTR exceeds that of ReLoop2. It is understandable since, with the passage of time, new data distribution changes dramatically from the initial data distribution, and as a result, the accuracy of the original base model's prediction for new data decreases. As ReLoop2 relies on base model prediction and error memory with no training process, it is likely that the AUC of IncCTR exceeds that of ReLoop2 when the time slot increases. From another point of view, additional training, like incremental learning, is necessary since it can make the model have better control over new data by updating the model parameters. By combining IncCTR and ReLoop2, DCNv2 achieves the best performance in Figure 4, demonstrating the efficiency of our fast model adaptation module.

4.4 Ablation Studies

4.4.1 Effect of K for Memory Reading. We investigate the effect of K in Figure 5. When K is small, error compensation is mainly determined by a small number of neighbors, which can not stand for the average error in the error memory, leading to a higher but not the best gAUC. When K is too large, the final output will be influenced by those neighbor samples that are not so similar to itself, resulting in a slight decrease in gAUC, but it is generally stable.
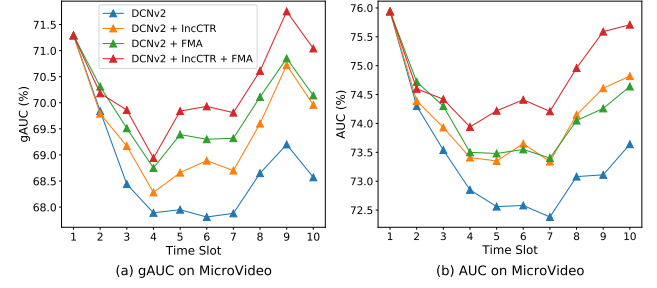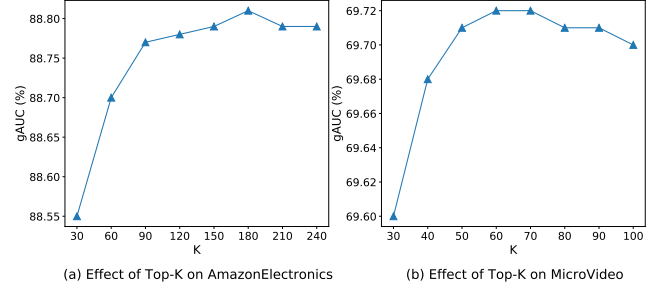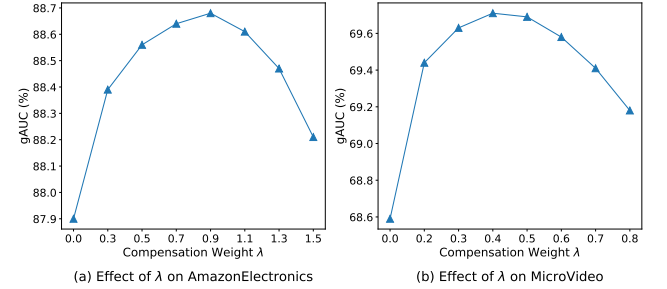
The best results are obtained when we choose an appropriate K. Through our experiment, we find that K=180 and K=70 achieve the best performance of DCNv2 on AmazonElectronics and MicroVideo, respectively.

4.4.2 Effect of Compensation Weight λ. We investigate the effect of compensation weights λ in Figure 6. When λ is small, the final output is mainly determined by the base model output, thus a bit higher but relatively close to the baseline. Specifically, when $\lambda = 0$, the final output is the same as that of the base model, serving as the baseline. When λ is too large, error compensation contributes more to the final output. gAUC of the final output decreases because of the lower percentage of base model, whose accuracy is supported by a large amount of training data. We empirically find $\lambda = 0.9$ and $\lambda = 0.4$ achieve the best performance of DCNv2 on AmazonElectronics and MicroVideo, respectively.

5 RELATED WORK

CTR Prediction. CTR prediction plays a key role in online advertising, recommender system, and information retrieval. Even a small improvement in CTR prediction can have a significant impact, benefiting both users and platforms. As a result, extensive research efforts have been dedicated to this field, both in academia and industry. In general, the goal of CTR prediction is to generate probability scores that represent user preferences for item candidates in specific contexts. Recently, a plethora of CTR prediction approaches have been proposed, ranging from traditional logistic regression (LR) models [45], factorization machines (FM) models [24, 53], to various deep neural network (DNN) models. Many of these models focus on designing feature interaction operators to capture complex relationships among features, such as product operators [17, 51, 55], bilinear interaction and aggregation [20, 43], factorized interaction layers [70], convolutional operators [34, 38, 40], and attention operators [5, 54]. Additionally, user behavior sequences play a crucial role in modeling user interests. Different models have been employed for behavior sequence modeling, including attention-based models [66, 67], memory network-based models [48, 52], and retrieval-based models [49, 50]. Notably, the BARS benchmark [68, 69] provides a comprehensive review and benchmarking results of existing CTR prediction models. However, all of these models focus on modeling sequential patterns during training and assume fixed parameters during testing, making them incapable of handling distribution shifts.

Incremental Learning. Incremental learning is a general framework that aims to continuously update model parameters to acquire new knowledge from new data while preserving the model's ability to generalize on old data [15]. In the context of recommender systems, incremental training has been widely adopted to cope with the data distribution shift and minimize the generalization gap between training and testing. Typically, model parameters from the previous version are reused as an initialization for the next round of training [26]. To alleviate the catastrophic forgetting problem, Wang et al. [61] proposed the IncCTR method, which uses knowledge distillation to strike a balance between retaining the previous pattern and learning from the new data distribution. In our earlier work, we introduced the ReLoop framework[3], which establishes a self-correcting learning loop during the model training phase. However, ReLoop2 focuses on test-time adaptation instead. Integrating both techniques is an interesting direction and we leave it for future research. Other studies [11, 47, 65] apply meta-learning techniques to incremental training of recommendation models, aiming to facilitate knowledge transfer from old parameters to new parameters. A recent study [39] proposes an adaptive incremental learning algorithm for mixture-of-experts (MoE) models to adapt to concept drift. Instead, our work is orthogonal to incremental training and focuses on enabling fast model adaptation through error compensation using a non-parametric memory approach. Furthermore, in contrast to the majority of continual learning studies [57], ReLoop2 employs a refreshed error memory for model adaptation, deviating from the conventional practice of utilizing a memory buffer for experience replay to prevent catastrophic forgetting.

Retrieval Augmentation. Our work also draws some inspiration from recent research on retrieval augmented machine learning techniques [64]. Retrieval augmentation focuses on retrieving similar key-value pairs from the external memory to enhance model generalizability, particularly for rare events or long-tail classes [25]. This approach has been successfully applied in various domains, including neural machine translation [28, 46, 60], visual recognition [22, 41], question answering [31], retrieval-augmented pre-training [14, 19] and text-to-image generation [2]. However, unlike these studies that retrieve data for model training, our work leverages refreshed online data for retrieval-augmented model adaptation. Additionally, we present a time- and memory-efficient design for top-k retrieval in large-scale online recommendation scenarios.

6 CONCLUSION

In this paper, we make a pioneering effort towards fast adaptation of CTR prediction models for online recommendation. To address the challenge of distribution shifts in streaming data, we introduce a slow-fast learning paradigm inspired by the complementary learning systems observed in human brains. In line with this paradigm, we propose ReLoop2, a self-correcting learning loop that facilitates fast model adaptation in online recommender systems through responsive error compensation. Central to ReLoop2 is a non-parametric error memory module that is designed to be time- and space-efficient and undergoes continual refreshing with newly observed data samples during model serving. Through comprehensive experiments conducted on open benchmark datasets and our production dataset, we demonstrate the effectiveness of ReLoop2 in enhancing model adaptiveness under distribution shifts.

ACKNOWLEDGMENTS

REFERENCES

[1] Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. 2022. Learning Fast, Learning Slow: A General Continual Learning Method based on Complementary Learning System. In *The Tenth International Conference on Learning Representations (ICLR)*.

[2] Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. 2022. Retrieval-Augmented Diffusion Models. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[3] Guohao Cai, Jieming Zhu, Quanyu Dai, Zhenhua Dong, Xiuqiang He, Ruiming Tang, and Rui Zhang. 2022. ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2692–2697.

[4] Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Annual ACM Symposium on Theory of Computing (STOC)*. 380–388.

[5] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. *CoRR* abs/1905.06874.

[6] Xusong Chen, Dong Liu, Zheng-Jun Zha, Wengang Zhou, Zhiwei Xiong, and Yan Li. 2018. Temporal Hierarchical Attention at Category- and Item-Level for Micro-Video Click-Through Prediction. In *2018 ACM Multimedia Conference on Multimedia Conference (MM)*. 1146–1153.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys)*. 7–10.

[8] Benjamin Coleman and Anshumali Shrivastava. 2020. Sub-linear RACE Sketches for Approximate Kernel Density Estimation on Streaming Data. In *The Web Conference 2020 (WWW)*. 1739–1749.

[9] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of 25th International Conference on Very Large Data Bases (VLDB)*. 518–529.

---

[10] Michel X. Goemans and David P. Williamson. 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM* 42, 6 (1995), 1115–1145.

[11] Renchu Guan, Haoyu Pang, Fausto Giunchiglia, Ximing Li, Xuefeng Yang, and Xiaoyue Feng. 2022. Deployable and Continuable Meta-learning-Based Recommender System with Fast User-Incremental Updates. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1423–1433.

[12] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[13] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, Vol. 119. 3887–3896.

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 119)*. 3929–3938.

[15] Jiangpeng He, Runyu Mao, Zeman Shao, and Fengqing Zhu. 2020. Incremental Learning in Online Scenario. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 13923–13932.

[16] Ruining He and Julian J. McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*. 507–517.

[17] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

[18] Monika Rauch Henzinger. 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 284–291.

[19] Ziniu Hu, Ahmet Iscen, Chen Sun, Zirui Wang, Kai-Wei Chang, Yizhou Sun, Cordelia Schmid, David A. Ross, and Alireza Fathi. 2022. REVEAL: Retrieval-Augmented Visual-Language Pre-Training with Multi-Source Multimodal Knowledge Memory. *CoRR* abs/2212.05221 (2022).

[20] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of ACM Conference on Recommender Systems, (RecSys)*. 169–177.

[21] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*. 604–613.

[22] Ahmet Iscen, Alireza Fathi, and Cordelia Schmid. 2023. Improving Image Recognition by Retrieving from Web-Scale Image-Text Data. *CoRR* abs/2304.05173 (2023).

[23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.

[24] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (Recsys)*. ACM, 43–50.

[25] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to Remember Rare Events. In *The 5th International Conference on Learning Representations (ICLR)*.

[26] Petros Katsileros, Nikiforos Mandilaras, Dimitrios Mallis, Vassilis Pitsikalis, Stavros Theodorakis, and Gil Chamiel. 2022. An Incremental Learning framework for Large-scale CTR Prediction. In *Sixteenth ACM Conference on Recommender Systems (RecSys)*. 490–493.

[27] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest Neighbor Machine Translation. In *Proceedings of the 9th International Conference on Learning Representations (ICLR)*.

[28] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest Neighbor Machine Translation. In *9th International Conference on Learning Representations (ICLR)*.

[29] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*.

[30] Dharshan Kumaran, Demis Hassabis, and James L McClelland. 2016. What Learning Systems Do Intelligent Agents Need? Complementary Learning Systems Theory Updated. *Trends in Cognitive Sciences* 20(7) (2016), 512–534.

[31] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[32] Ruirui Li, Xian Wu, Xiusi Chen, and Wei Wang. 2020. Few-Shot Learning for New User Recommendation in Location-based Social Networks. In *The Web Conference 2020 (WWW)*. 2472–2478.

[33] Yongqi Li, Meng Liu, Jianhua Yin, Chaoran Cui, Xin-Shun Xu, and Liqiang Nie. 2019. Routing Micro-videos via A Temporal Graph-guided Recommendation System. In *Proceedings of the 27th ACM International Conference on Multimedia, (MM)*. ACM, 1464–1472.

[34] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, (CIKM)*. ACM, 539–548.

[35] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, et al. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (KDD)*. 1754–1763.

[36] Jian Liang, Ran He, and Tieniu Tan. 2023. A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts. *CoRR* abs/2303.15361 (2023).

[37] Guoliang Lin, Hanlu Chu, and Hanjiang Lai. 2022. Towards Better Plasticity-Stability Trade-off in Incremental Learning: A Simple Linear Connector. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 89–98.

[38] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, (WWW)*. 1119–1129.

[39] Congcong Liu, Yuejiang Li, Xiwei Zhao, Changping Peng, Zhangang Lin, and Jingping Shao. 2022. Concept Drift Adaptation for CTR Prediction in Online Advertising Systems. *CoRR* abs/2204.05101 (2022).

[40] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, (CIKM)*. ACM, 1743–1746.

[41] Alexander Long, Wei Yin, Thalaiyasingam Ajanthan, Vu Nguyen, Pulak Purkait, Ravi Garg, Alan Blair, Chunhua Shen, and Anton van den Hengel. 2022. Retrieval Augmented Classification for Long-Tail Visual Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6949–6959.

[42] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[43] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[44] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. 1995. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review* 102(3):419 (1995).

[45] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1222–1230.

[46] Yuxian Meng, Xiaoya Li, Xiayu Zheng, Fei Wu, Xiaofei Sun, Tianwei Zhang, and Jiwei Li. 2022. Fast Nearest Neighbor Machine Translation. In *Findings of the Association for Computational Linguistics (ACL)*. 555–565.

[47] Danni Peng, Sinno Jialin Pan, Jie Zhang, and Anxiang Zeng. 2021. Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems. In *Fifteenth ACM Conference on Recommender Systems (RecSys)*. 411–421.

[48] Qi Pi, Weijie Bian, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Practice on Long Sequential User Behavior Modeling for Click-Through Rate Prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. ACM, 2671–2679.

[49] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based User Interest Modeling with Lifelong Sequential Behavior Data for Click-Through Rate Prediction. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. 2685–2692.

[50] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*. 2347–2356.

[51] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-Based Neural Networks for User Response Prediction. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM)*. 1149–1154.

[52] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, and Kun Gai. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 565–574.

[53] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. 995–1000.

[54] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. 1161–1170.

[55] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference (WWW)*. 2828–2837.

[56] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *International Conference on Management of Data (SIGMOD)*. 2614–2627.

[57] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *CoRR* abs/2302.00487 (2023).

[58] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 11th International Workshop on Data Mining for Online Advertising (ADKDD)*. 12:1–12:7.

[59] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference (WWW)*. 1785–1797.

[60] Shuhe Wang, Jiwei Li, Yuxian Meng, Rongbin Ouyang, Guoyin Wang, Xiaoya Li, Tianwei Zhang, and Shi Zong. 2021. Faster Nearest Neighbor Machine Translation. *CoRR* abs/2112.08152 (2021).

[61] Yichao Wang, Huifeng Guo, Ruiming Tang, Zhirong Liu, and Xiuqiang He. 2020. A Practical Incremental Method to Train Deep CTR Models. *CoRR* abs/2009.02147 (2020).

[62] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2021. Generalizing from a Few Examples: A Survey on Few-shot Learning. *ACM Comput. Surv.* 53, 3 (2021), 63:1–63:34.

[63] Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory Networks. In *3rd International Conference on Learning Representations (ICLR)*.

[64] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-Enhanced Machine Learning. In *The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2875–2886.

[65] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System?: A Sequential Meta-Learning Method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 1479–1488.

[66] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *CoRR* abs/1809.03672 (2018).

[67] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1059–1068.

[68] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *The 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 2759–2769.

[69] Jieming Zhu, Kelong Mao, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Zhicheng Dou, Xi Xiao, and Rui Zhang. 2022. BARS: Towards Open Benchmarking for Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 2912–2923.

[70] Jieming Zhu, Guohao Cai Qinglin Jia, Quanyu Dai, Jingjie Li, Zhenhua Dong, Ruiming Tang, and Rui Zhang. 2023. FINAL: Factorized Interaction Layer for CTR Prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.