RelayRec: Empowering Privacy-Preserving CTR Prediction via Cloud-Device Relay Learning

Yongheng Deng[1], Guanbo Wang[1], Sheng Yue[1], Wei Rao[2], Qin Zu[2],
Wenjie Wang[2], Shuai Chen[2], Ju Ren[1,3,*], Yaoxue Zhang[1,3]
[1]Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing, China
[2]Meituan, Beijing, China
[3]Zhongguancun Laboratory, Beijing, China
{dyh19@mails.,wanggb23@mails.,shengyue@,renju@,zhangyx@}tsinghua.edu.cn
{raowei,zuqin,wangwenjie22,chenshuai31}@meituan.com

ABSTRACT

Click-through rate (CTR) prediction holds paramount importance across numerous applications, profoundly impacting user experience and business profitability. The freshness of a CTR prediction model significantly influences its performance, since users' needs and interests may be changing over time, thereby requiring the model to be updated frequently. However, stringent data protection regulations have constrained the collection of users' personal data, posing challenges to traditional model refreshing strategies that rely on centralized data collection. On-device learning techniques, such as federated learning (FL), offer a viable solution by enabling model training on devices without compromising user privacy. Nevertheless, the scarcity of training data with diverse distributions among devices presents considerable obstacles to on-device learning effectiveness. To address these challenges, we introduce RelayRec, a cloud-device relay learning framework designed for privacy-preserving CTR prediction. To establish competent initial models for devices, RelayRec categorizes pre-regulation cloud data into user preference groups, training preference-specific models for devices. Furthermore, a cloud-based automated model selector is developed to identify suitable initial models for devices. To elevate the relay learning performance of these initial models, we incorporate a personalized collaborative learning mechanism that aggregates device models based on user preferences. Extensive experimental evaluations underscore RelayRec's superior performance compared to state-of-the-art benchmarks, affirming its efficacy in privacy-preserving CTR prediction.

1 INTRODUCTION

With the rapid growth of the Internet and the e-economy, numerous clicking surges in various online activities, such as shopping, video streaming, and web browsing. Click-through Rate (CTR) prediction, which aims to predict the probability of a user clicking or interacting with a recommended item, plays a vital role in many applications such as recommendation systems, online advertising, and Web search. It has been shown that even a tiny improvement in prediction accuracy can potentially result in a large enhancement on user experience and platform revenue [4, 29, 42]. Therefore, in both academia and industry, CTR prediction has attracted considerable research attentions in recent years. On the one hand, multifarious CTR prediction models are developed to boost the prediction accuracy, evolving from simple logistic regression (LR) [36], factorization machines (FM) [18], to deep neural networks (DNN) [5], Wide&Deep [4], DIN [48], etc. On the other hand, enormous volume of user behavior data is amassed in the cloud server by companies, which is used to train advanced CTR prediction models.

Despite the remarkable advancements achieved in CTR prediction, several bottlenecks still exist, impeding its practical efficacy within real-world applications. Notably, there is usually a substantial performance gap between the training and inference phases of a prediction model, as user's clicking behaviors typically vary over time due to the changes of their need or interest. As a result, models trained with previously collected user data may quickly become obsolete when deployed to serve users on devices, leading to unsatisfactory prediction performance. Therefore, the prediction models are required to be updated periodically to guarantee their freshness. However, the scope of data fields involved in CTR prediction include user profile and behavior information, e.g., user ID, gender, age, and clicked items, which are usually privacy-sensitive. With the increasingly heightened privacy awareness, users may refuse to share their private data. Furthermore, a multitude of data protection regulations such as GDPR have imposed stringent limitations on the collection of user data. Under the new circumstances of privacy protection, refreshing prediction models to provide promising performance for users becomes a formidable challenge.

To address the above challenges, *on-device learning* [15, 44] provides a feasible solution by enabling model updates on devices using their local data, which simultaneously refresh the model while ensuring user privacy by keeping private data locally. However, the individual training samples available on each device can be quite limited, making on-device learning suffer from local optimization and suboptimal performance [11, 47]. To mitigate this limitation, *federated learning (FL)* [30] proposes a collaborative model training paradigm for privacy-preserving systems. In FL, each device keeps its private data locally and trains a shared model with their local data on the device. To overcome the sample size limitation of individual devices, the model parameters of each device are transferred to the cloud server, which are aggregated to collaboratively update the shared model among devices. While FL has demonstrated remarkable progress in terms of learning and preserving privacy, there remains ample potential for optimizing its collaborative learning performance. Because the inherent divergence in user preferences results in non-independent and identically distributed (non-IID) training data across devices, which poses considerable hurdles to the convergence and efficiency of FL.

[*] Corresponding author.

979-8-3503-6201-5/24/$31.00 ©2024 IEEE
DOI 10.1109/IPSN61024.2024.00020

Under non-IID training data across devices, prior arts proposed several optimization algorithms, e.g., FedProx [27], SCAFFOLD [19], and FedNova [41], as derivatives of the FedAvg algorithm to improve the training performance of the shared model in FL. Another line of works improve the performance of non-IID FL by participant selection [7, 22, 40], which select devices with high-utilities to participate in training. Besides, personalized FL [21] aims to tailor a personalized model for each device instead of learning one shared model to fit all devices. Existing works mainly achieve model personalization via local fine-tuning [10, 38], clustering [12, 33, 37], knowledge distillation [2, 8, 26], pruning [6, 24, 25], etc. However, prior efforts have primarily concentrated on optimizing the collaborative model training process within FL, while overlooking the benefits of the training data collected in the cloud before data regulations. Such data can be taken advantage to provide a well-initialized device model for users and consequently enhancing the collaborative training process of FL.

To bridge this gap, we propose RelayRec, a cloud-device relay learning framework for privacy-preserving CTR prediction. RelayRec enhances on-device prediction performance by fully utilizing both the historical data collected in the cloud before data regulations and the real-time data decentralized on devices after data regulations to relay on training CTR prediction models. Specifically, RelayRec integrates three key components. To provide a well-initialized model for each device, a *clustered meta-learning* mechanism is introduced to the cloud server. Considering that the preferences of users may vary over time, it divides the cloud data into groups based on user's preference, and trains multiple preference-wise models using meta-learning as the initial models for devices. Then an *automated model selector* is trained to intelligently choose an appropriate initial model for devices based on their user preference. In this way, even if the preference of a user changes, the model selector can choose a useful initial device model for it based on its new preference, and the model can be further trained with its new data on devices to rapidly adapt to the evolving preference. To improve the collaborative learning performance of devices with their generated new data after data regulations, a *personalized collaborative learning* mechanism is adopted among devices. It aggregates the updated model parameters of devices based on their choice of initial models, which can promote beneficial collaborations between users with similar preferences.

We conduct extensive experiments on real-world CTR prediction tasks to evaluate the performance of RelayRec. Specifically, we compare its performance with state-of-the-art approaches and demonstrate its superior on-device prediction performance. Our evaluation encompasses detailed breakdown analyses that illustrate the effectiveness of each pivotal component within RelayRec. Furthermore, we conduct sensitivity analysis to investigate the impact of several important hyperparameters' settings on RelayRec.

Overall, the key contributions of this paper can be summarized as follows:

- We propose a novel cloud-device relay learning framework for privacy-preserving CTR prediction, which can take advantage of both pre-regulation cloud-collected historical data and post-regulation real-time data decentralized on devices to promote on-device prediction performance.
- We propose a clustered meta-learning mechanism to learn well-initialized device models, an automated model selector mechanism to choose appropriate initial models for devices, and a personalized collaborative learning mechanism to relay on updating models efficiently.
- We conduct extensive experiments with real-world and industrial datasets, demonstrating the superior performance of RelayRec compared to state-of-the-art methods and the effectiveness of the designed components.

Table 1: AUC comparison of a well-trained DNN prediction model before and after user preference changes.

	MovieLens	Frappe	KKBox	Meituan
Before	0.871	0.884	0.613	0.711
After	0.861	0.876	0.599	0.707
Reduction	↓ 1.15%	↓ 0.9%	↓ 2.28%	↓ 0.56%

2 BACKGROUND AND MOTIVATION

We start with an introduction on the background of CTR prediction, followed by the need for relay learning and the challenges of relay learning that motivate our work.

2.1 Background of CTR Prediction

The objective of CTR prediction is to predict the probability that a user will click a given item. Formally, we define the estimated click probability as $\hat{y} = \sigma(\phi(x))$, where x is the input features, $\phi(\cdot)$ is the CTR prediction model function, $\sigma(\cdot)$ is the sigmoid function that maps $\hat{y}$ to $[0, 1]$. Constructing the input features, building the prediction model, and learning the model parameters from training data, are key parts in CTR prediction modeling. Specifically, the input instances contain a number of feature fields, generally including user profile, item profile, and context information. These features are mostly very sparse and hence are usually mapped into low-dimensional dense vectors by feature embeddings. Then the dense vectors can be used in various CTR prediction models to capture the sophisticated high-order interactions between features for accurate predictions. Previous works have proposed diverse CTR prediction models, e.g., LR [36], FM [18], Wide&Deep [4], DeepFM [13], DCN [42], xDeepFM [28], which have been successfully applied in industry. To learn the model parameters, the binary cross-entropy loss is widely utilized:

$$\mathcal{L}(\mathcal{D}) = -\frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}}(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i)), \tag{1}$$

where $\mathcal{D}$ is the training dataset, y_i and $\hat{y}_i$ represent the ground truth and the estimated click probability of sample i.

2.2 The Need for Relay Learning

In the industry, it's common to gather extensive user data for training CTR prediction models. Besides, the learned models often require iterative updates to maintain their freshness, as user's needs and interests tend to change over time. For example, users' needs and interests may shift from seeking summer clothing options like dresses and shorts to winter clothing items like sweaters and coats as the seasons change. The freshness of a CTR prediction

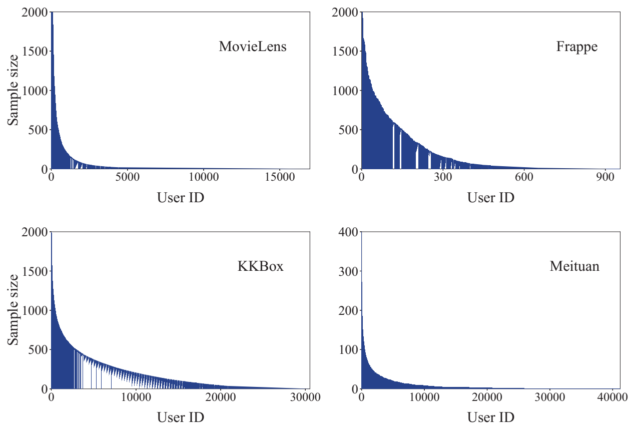

Figure 1: Statistics of user's sample size in real-world datasets.

model is imperative to the prediction performance on users. To show a data point, we compare the prediction performance of a well-trained DNN model before and after user preference changes. From the results in Table 1, we can observe that the model suffers considerably when user's preference shifts. (The experimental settings and dataset descriptions will be detailed in Section 5.1.1.) To mitigate this issue, a practical solution is to collect users' new interaction logs continuously, re-train the prediciton models and deliver a new model version for devices evolutionarily. However, this model refreshing method involves prominent privacy concerns because of the ongoing collection of user feedbacks, which often contains sensitive information. Therefore, to refresh model in a privacy-preserving manner, on-device learning attracts growing attentions in recent years. After the imposition of data privacy regulations, the new click feedbacks of users are no longer collected to the cloud server, instead they are retained on devices and the model is updated with them locally. In this way, the model is first trained on the cloud server with the collected data before data regulations and then relay-trained on devices after data regulations. As such, prediction models on devices are refreshed without exposing user's private interactions.

2.3 Challenges of relay learning

The relay learning paradigm offers a privacy-preserving avenue for refreshing models, however, enhancing CTR prediction performance on devices presents significant challenges. First, the number of training samples per user is typically limited, as exemplified by the sample size statistics shown in Fig. 1 for four real-world datasets. Many users have severely restricted sample sizes available for training. To address the data constraints inherent to individual devices and harness the collective potential of multiple devices, FL proposes to collaboratively learn a shared model among multiple devices without sharing their private data. In FL, each participating device downloads a global model from the cloud server, trains the model locally with their own data, and uploads the trained model parameters to the cloud server for aggregation. The cloud server updates the global model by aggregating the received device models and then distributes the updated global model to devices for the next round of iteration.

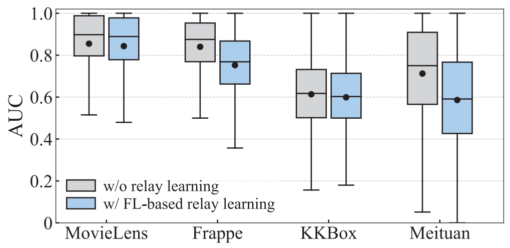

Figure 2: Performance comparison of on-device prediction without relay learning and with FL-based relay learning.

The learning paradigm of FL takes advantage of private data from numerous devices to train the global model. However, it is not straightforward to apply FL to CTR prediction. For example, the performance of the widely used FedAvg [30] algorithm in FL, as evaluated on four datasets (shown in Fig. 2), falls significantly short of expectations and can even degrade on-device inference performance. This underwhelming performance of FL can be attributed to two primary aspects. On the one hand, individual devices possess limited local training samples, which hinders their individual training efficacy. Moreover, effective collaboration between devices is pivotal, and the non-IID characteristics in local data distribution across devices, stemming from the varying user preferences, present substantial challenges to efficient device collaboration [6, 17]. While some research efforts have addressed the non-IID issue in FL [22, 27, 33], few have focused on relay learning between the cloud server and devices.

In relay learning, in addition to the challenges associated with collaborative learning between devices under non-IID training data, how to efficiently utilize the centrally collected data on the cloud server before data regulations also has a significant impact on the prediction performance. The design of RelayRec focuses on addressing these pivotal challenges of cloud-device relay learning to improve the model training performance.

3 MAIN IDEA AND SYSTEM OVERVIEW

This section describes the main idea of our relay learning solutions and gives an overview on the architecture of RelayRec.

3.1 Main Idea

Since the cloud server has some data collected before data privacy regulations, RelayRec aims to effectively exploit these data for the initial training of device models on the cloud server, in order to provide well-initialized models for the relay learning on devices. A straightforward approach is to train a device model with these data as the initial learning model for all devices. However, it has been shown in Section 2.2 that the performance of the model trained in this manner is less favourable when the preferences of users vary over time. Besides, based on the initial model learned in this manner, the on-device learning performance is also suboptimal as demonstrated in Section 2.3. We delve into the problem of unsatisfactory model performance and identify two primary underlying causes. One the one hand, when the local data distributions of devices evolve with user preferences, the model will learn new knowledge from scratch. As a result, it cannot fully benefit from the initial model provided by the cloud server. On the other hand, limited

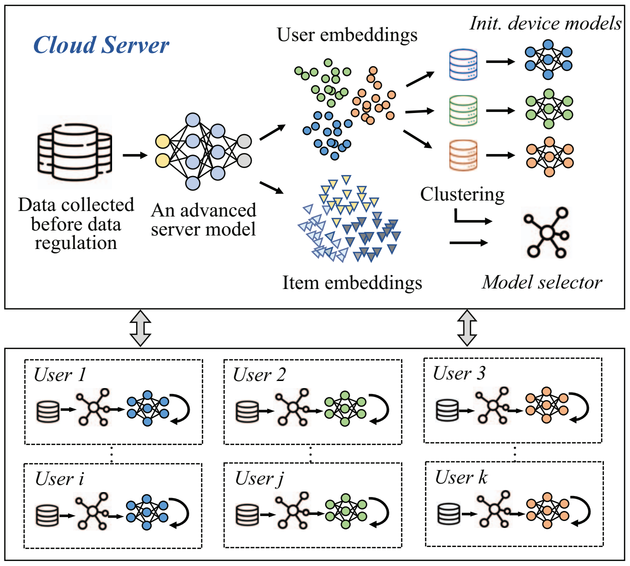

Figure 3: System overview of RelayRec.

by the scarcity of local training samples on individual devices, on-device training is susceptible to local optimization. To address these issues simultaneously, we introduce a clustered meta-learning approach along with an automated model selector mechanism. Specifically, the device model is trained with meta-learning in the cloud server, such that the model can adapt to user's varying preference rapidly with limited new data generated on devices. Moreover, instead of learning a single initial device model in the cloud server, we divide the cloud data into groups based on user preferences, and learn seperate device models for each group to derive multiple initial device models. Then, a model selector network is trained to automatically choose an appropriate initial model for each device. In this way, even if a user's preferences change, the model selector can choose an appropriate initial model for it based on the updated preference, so that the on-device learning can still benefit from the previously acquired knowledge of the model. In contrast, newly generated user data is distributed on devices after data regulations. Each device downloads an appropriate initial model from the cloud server, and further trains it with their new data collaboratively. To improve the collaborative learning performance between users, we employ a personalized collaboration approach in RelayRec. Specifically, instead of aggregating the device model parameters from all users, model aggregation is conducted based on the model selection results, i.e., the updated model parameters of devices that downloads the same initial model are aggregated separately. The design of RelayRec is introduced in the next sections.

3.2 System Overview

The architecture of RelayRec is illustrated in Fig. 3, which is a relay learning framework between the cloud server and devices. In the cloud server, data collected before data regulations is employed to train initial device models and a model selector network for devices. To achieve this, RelayRec introduces generic methods that are compatible with various CTR prediction tasks. Specifically, an advanced server model is first trained with the cloud data to derive embeddings of the user-related data fields and the item-related data fields. Then, users are clustered into groups based on the similarities of their user embeddings. Within each cluster, a lightweight device model is trained via meta-learning utilizing the data collected from the users within the group. On the other hand, the item embeddings together with the user clustering results are used to construct training samples for the model selector. Afterwards, each device first downloads the well-trained model selector from the cloud server. The new data generated after data regulations on each device is input to the model selector, and the model selector will output the recommended initial device model ID. Based on the ID, each device downloads the corresponding initial device model from the cloud server, and proceeds to enhance it with the generated local new data. Finally, the updated device model parameters are uploaded to the cloud server, where they are aggregated separately based on device's initial model. These aggregated parameters are then sent back to devices for the next round of iteration.

4 SYSTEM DESIGN

This section gives a detailed description on the core components of RelayRec. We begin by introducing the clustered meta-learning mechanism, followed by the automated model selector and personalized collaborative learning mechanisms.

4.1 Clustered Meta-Learning Mechanism

Given that a bunch of user data is collected to the cloud server before data regulations, RelayRec aims to make full use of these data to learn well-initialized device models for users. Formally, we represent the data collected to the cloud server as $\mathcal{D}_s = \cup_{u \in \mathcal{U}_s} \mathcal{D}_u$, where $\mathcal{D}_u$ is user u's data generated before data regulations. After data regulations, the new data $\hat{\mathcal{D}}_u$ generated by each user $u \in \mathcal{U}$ is kept on their local device. A basic idea is to train a CTR prediction model at the cloud server with $\mathcal{D}_s$ to derive an initial parameter $\boldsymbol{w}^0$, and then share $\boldsymbol{w}^0$ among all users $\mathcal{U}$. However, since the size of $\hat{\mathcal{D}}_u$ is typically small, $\boldsymbol{w}^0$ is easy to suffer from local optimization when further trained with $\hat{\mathcal{D}}_u$. Moreover, the distribution of the newly generated data $\hat{\mathcal{D}}_u$ may vary from $\mathcal{D}_u$ due to the preference changes of users, and the users in $\mathcal{U}$ may be different from $\mathcal{U}_s$ with the emergence of new users. To address these issues, meta-learning seems to be a potential solution [9, 23, 34], as it aims to train a model that can rapidly adapt to new tasks with a few samples [39]. The widely used optimization-based meta-learning algorithm considers a model and a distribution over task, aiming to find desirable parameter of the model. It starts from a random initialized parameter, samples mutiple tasks from the task distribution, and performs local and global updates to optimize the parameter. Through local updates, the parameter is updated by the gradient for each sampled task. Following these local updates, the algorithm performs global updates based on the test loss on each task with the updated parameter. This iterative process ensures that the parameter after global updates becomes better suited to handling diverse tasks and effectively adapting to new ones.

Adopting meta-learning, making CTR prediction for each user $u \in \mathcal{U}_s$ can be regarded as a task τ_u. Each task τ_u is divided into a

Algorithm 1: The clustered meta-learning algorithm of RelayRec.

```
Input: cloud data $\mathcal{D}_s = \cup_{u \in \mathcal{U}_s} \mathcal{D}_u$; user cluster number $K$;
       advanced server model $f_s$ with randomly initialized
       parameter $\boldsymbol{w}_s$; device model $f$; step size $\eta, \eta_1, \eta_2$.
// Train advanced server model with cloud data
1  $\boldsymbol{w}_s \leftarrow \boldsymbol{w}_s - \eta \nabla_{\boldsymbol{w}_s} \mathcal{L}(\mathcal{D}_s; f_s(\boldsymbol{w}_s))$;
// Calculate user embeddings with $f_s^e$
2  $\boldsymbol{e}_u \leftarrow f_s^e(u; \boldsymbol{w}_s)$ foreach $u \in \mathcal{U}_s$;
// User clustering based on user embeddings
3  $\{\mathcal{U}_s^1, \mathcal{U}_s^2, \ldots, \mathcal{U}_s^K\} \leftarrow$ k-means$(\{\boldsymbol{e}_u\}_{u \in \mathcal{U}_s})$;
4  for $k = 1, 2, \ldots, K$ do
       // Meta-learning
5      randomly initialize $\boldsymbol{w}_k^0$;
6      while not converge do
7          Sample a batch of users $B \sim p(\mathcal{U}_s^k)$;
8          foreach $u \in B$ do
9              Set $\theta_u \leftarrow \boldsymbol{w}_k^0$;
10             Calculate $\nabla_{\theta_u} \mathcal{L}(\mathcal{D}_u^{sup}; f(\theta_u))$;
11             Local update $\theta'_u \leftarrow \theta_u - \eta_1 \nabla_{\theta_u} \mathcal{L}(\mathcal{D}_u^{sup}; f(\theta_u))$;
12         Global update
           $\boldsymbol{w}_k^0 \leftarrow \boldsymbol{w}_k^0 - \eta_2 \sum_{u \in B} \nabla_{\boldsymbol{w}_k^0} \mathcal{L}(\mathcal{D}_u^{query}; f(\theta'_u))$;
13     Finetune $\boldsymbol{w}_k^0$ using $\mathcal{D}_s$;
14 return $\{\boldsymbol{w}_0^0, \boldsymbol{w}_1^0, \ldots, \boldsymbol{w}_K^0\}$, $\{\mathcal{U}_s^0, \mathcal{U}_s^1, \ldots, \mathcal{U}_s^K\}$;
```

support set $\mathcal{D}_u^{sup}$ and a query set $\mathcal{D}_u^{query}$, which are used to calculate the training loss and test loss on each task respectively. Specifically, the support set is utilized during the local update phase, where the algorithm adjusts the model's parameter based on each support set to facilitate the learning process. In contrast, the query set is employed in the global update phase, where the algorithm trains the parameter to minimize losses using the adapted parameters from the query sets, enabling a learning-to-learn process. Through this process, the model can efficiently adapt to new tasks. However, prior literature revealed that most meta-learning approaches are difficult to achieve optimal performance with a single initialization when the distribution of tasks varies a lot [35, 45]. Therefore, since users have substantially different perference distributions, it is unwise to learn a common initial meta-model for all users. Such approaches can make the learning process unsmooth and hinder model convergence, impeding its adaptability to new tasks [35, 46].

To handle the above issue, we propose a clustered meta-learning mechanism. The core idea of this approach is clustering users into groups based on their preferences, so that the users within the same group have similar preferences, resulting in similar distributions of tasks for meta-learning. To achieve this, RelayRec adopts an embedding-based method to achieve clustering, which is relatively generic for various CTR prediction tasks. Specifically, the cloud data is first utilized to train an advanced CTR prediction model, in order to derive embeddings of the input data. Then user clustering is performed based on their user embeddings, which consists of the embeddings of the user-related feature fields in the input data, e.g., user ID, gender, and age. In this way, the cloud data is divided into groups based on the user clustering results. In each group, an initial device model is learned with the data corresponding to the users within the group.

Algorithm 1 shows the details of the clustered meta-learning method. First, the advanced server model f_s is trained with the cloud data $\mathcal{D}_s$ to derive the user embeddings. Then the k-means algorithm is performed on the calculated user embeddings and get the user clustering results $\{\mathcal{U}_s^0, \mathcal{U}_s^1, \ldots, \mathcal{U}_s^K\}$. Afterwards, meta-learning is conducted in each cluster k to learn an initial parameter $\boldsymbol{w}_k^0$ of device model f. Specifically, the model randomly samples a batch of users within the cluster and locally updates the parameters θ_u for each user u in the batch with its support dataset $\mathcal{D}_u^{sup}$. Then the algorithm globally updates $\boldsymbol{w}_k^0$ based on the loss for the sampled users' query dataset $\mathcal{D}_u^{query}$ with the updated parameter θ'_u. The local and global updating process will repeat until $\boldsymbol{w}_k^0$ converges. Finally, $\boldsymbol{w}_k^0$ is fine-tuned with the cloud data to get the final initial parameters for devices. The intuition of the fine-tuning process is two-fold: on the one hand, it is shown in previous work [34] that the representation and generalization abilities of the model learned by meta-learning are weakened significantly due to the data limitation of a single task. Therefore, we fine-tune the model with a large amount of cloud data to enhance the representation and generalization performance; one the other hand, if the initial model is directly downloaded and fine-tuned on devices, the performance will be limited by the constrained data size of a single device. In light of these, we fine-tune the initial model with the available cloud data in order to provide better initial models for devices.

4.2 Automated Model Selector Mechanism

With the clustered meta-learning algorithm, multiple initial models are derived at the cloud server, then devices can download one of the initial model and adapt it to user's new data locally. To choose an appropriate initial model for each device, a model selector is required. One may consider assigning initial model for devices based on the user clustering results at the cloud server, however, this is suboptimal due to the evolving of user preferences and the emergence of new users. Therefore, the model selector is required to dynamically choose an appropriate initial model based on user's timely preference. To achieve this, a straightforward and seemingly optimal approach is to allow devices to download all initial models from the cloud server, evaluate them using the local data, and choose the best-performing one. However, this method is evidently inefficient in practical applications. Therefore, we aim to learn a model selector network that is used on devices to choose an optimal initial model based on user's current preference. To this end, we propose an automated model selector mechanism, which learns a model selector network at the cloud server and applies it on devices to automatically choose the most suitable initial model. The model selector network is desired to take user's preference as input and outputs the ID of the recommended initial model learned at the cloud server.

To learn a desirable model selector network, RelayRec constructs training samples utilizing the cloud data and the user clustering results at the cloud server. To model user preferences, an intuitive approach is to leverage user profile information, e.g., user ID, gender, age. However, this approach can hamper the performance of

Algorithm 2: The automated model selector mechanism of RelayRec.

```
Input: cloud data $\mathcal{D}_s = \cup_{u \in \mathcal{U}_s} \mathcal{D}_u$; item cluster number $C$;
       server model with pre-trained parameter $\boldsymbol{w}_s$; total
       items set $\mathcal{I}$; user clustering results $\{k_u\}_{u \in \mathcal{U}_s}$; model
       selector network $s$ with randomly initialized
       parameter $\varphi$; step size $\eta_3$.
// Calculate item embeddings
1  $\boldsymbol{e}_i \leftarrow f_s^e(i; \boldsymbol{w}_s)$ foreach $i \in \mathcal{I}$;
// Item clustering based on item embeddings
2  $\{\mathcal{I}^1, \mathcal{I}^2, \dots, \mathcal{I}^C\} \leftarrow$ k-means$(\{\boldsymbol{e}_i\}_{i \in \mathcal{I}})$;
// Calculate cluster centroid embeddings
3  $\boldsymbol{e}_c \leftarrow \frac{1}{|\mathcal{I}_c|} \sum_{i \in \mathcal{I}_c} \boldsymbol{e}_i$ foreach $c = 1, 2, \dots, C$;
// User preference modeling
4  foreach $u \in \mathcal{U}_s$ do
5      $\lambda_c \leftarrow 0$ foreach $c = 1, 2, \dots, C$;
6      foreach $i \in \mathcal{D}_u$ do
7          $c \leftarrow$ find $c$ with maximum $sim(\boldsymbol{e}_i, \boldsymbol{e}_c)$;
8          $\lambda_c \leftarrow \lambda_c + 1$;
9      $\boldsymbol{p}_u \leftarrow \frac{1}{I_u} \sum_{c=1}^{C} \lambda_c \boldsymbol{e}_c$;   // $I_u$ is the number of
           interacted items of user $u$
// Construct training samples for model selector
10 Construct $\mathcal{D}_m \leftarrow \{(\boldsymbol{p}_u, k_u)\}_{u \in \mathcal{U}_s}$;
// Train the model selector network
11 $\varphi \leftarrow \varphi - \eta_3 \nabla_\varphi \mathcal{L}_{CE}(\mathcal{D}_m; s(\varphi))$;
12 return $s(\varphi)$, $\{\boldsymbol{e}_c\}_{c=1}^{C}$;
```

the model selector for new users with unestablished profiles due to the incompleteness of the training data. Compared to user profile information, the profile information of items is public and more readily accessible by the cloud server. Besides, the scenario considered in our paper involves a fixed set of items. Therefore, we advocate to model user preference based on their interacted items. However, the challenge confronted by this approach lies in that the number of items interacted by each user is different, making it unsuitable as direct input for the model selector network. Our solution to this issue involves categorizing all items into a fixed number of classes, and determining each user's preference based on the categories of items they have interacted.

Details of the automated model selector mechanism of RelayRec are described in Algorithm 2. Specifically, leveraging the pre-trained advanced server model f_s, we can derive the embeddings of all items $\mathcal{I}$. Then we perform clustering on the item embeddings and clusters $\mathcal{I}$ into C categories. In this way, we enable item categorizing without the need for manual analysis of item attributes, which is generic for diverse applications. For each category of items $\mathcal{I}_c$, we average their embeddings and get a cluster centroid embedding $\boldsymbol{e}_c$, indicating the features of each item category c. Then, for each interacted item $i \in \mathcal{D}_u$ of user u, we determine its category by calculating the similarity between its embeddings $\boldsymbol{e}_i$ and each cluster centroid embeddings $\boldsymbol{e}_c$. Based on the categories of user's interacted items, we utilize $\boldsymbol{p}_u \leftarrow \frac{1}{I_u} \sum_{c=1}^{C} \lambda_c \boldsymbol{e}_c$ to model the preference of user u. With user preference modeling, the training dataset for the model selector network can be constructed as $\mathcal{D}_m = \{(\boldsymbol{p}_u, k_u)\}_{u \in \mathcal{U}_s}$, where

Algorithm 3: The personalized collaborative learning mechanism of RelayRec.

```
Input: new data of users $\{\hat{\mathcal{D}}_u\}_{u \in \mathcal{U}}$; sampled user number
       per round $N$; local epoch $\tau$; model selector $s$ with
       pre-trained parameter $\varphi$; cluster centroid
       embeddings $\{\boldsymbol{e}_c\}_{c=1}^{C}$; step size $\eta$.
1  foreach $u \in \mathcal{U}$ do
2      Calculate $\boldsymbol{p}_u$ based on $\hat{\mathcal{D}}_u$ and $\{\boldsymbol{e}_c\}_{c=1}^{C}$;
3      $k_u \leftarrow s(\boldsymbol{p}_u; \varphi)$;      // Initial model selection
4      $\boldsymbol{w}_u \leftarrow$ Download initial device model $f(\boldsymbol{w}_{k_u}^0)$ from the
         cloud server;
5  foreach round $r = 1, 2, \dots$ do
6      $\mathcal{U}_p^r \leftarrow$ randomly sample $N$ users from $\mathcal{U}$;
7      foreach $u \in \mathcal{U}_p^r$ do in parallel
8          foreach local epoch $e = 1, 2, \dots, \tau$ do
               // On-device training with FedProx [27]
9              $\mathcal{L}_{FedProx} \leftarrow \mathcal{L}(\hat{\mathcal{D}}_u; f(\boldsymbol{w}_u)) + \frac{\mu}{2} ||\boldsymbol{w}_u - \boldsymbol{w}_{k_u}||^2$;
10             $\boldsymbol{w}_u \leftarrow \boldsymbol{w}_u - \eta \nabla_{\boldsymbol{w}_u} \mathcal{L}_{FedProx}$;
11         Upload $\boldsymbol{w}_u$ to the cloud server;
       /* Cloud server process */
12     foreach $k = 1, 2, \dots, K$ do
13         $\mathcal{U}_p^k \leftarrow \{u | k_u = k, \forall u \in \mathcal{U}_p^r\}$;
           // Clustered model aggregation
14         $\boldsymbol{w}_k \leftarrow \frac{1}{\sum_{u \in \mathcal{U}_p^k} |\hat{\mathcal{D}}_u|} \sum_{u \in \mathcal{U}_p^k} |\hat{\mathcal{D}}_u| \boldsymbol{w}_u$;
           // Update device model
15         Update $\boldsymbol{w}_u \leftarrow \boldsymbol{w}_k$ foreach $u \in \mathcal{U}_p^k$;
```

k_u is the clustering result of user u, corresponding to an initial model ID. Finally, the training task for the model selector network can be viewed as a classification task. We employ a Multi-Layer Perceptron (MLP) network as the model selector network s and train it with the cross-entropy loss on $\mathcal{D}_m$.

The model selector network is trained on the cloud server and downloaded by each device for local inference. On devices, the inputs to the model selector network are first constructed based on their newly generated interactions $\hat{\mathcal{D}}_u$, which aligns with the training data construction process of the model selector. Then the model selector outputs a cluster ID and the device will download the corresponding initial model from the cloud server. In this way, each user acquires a favorable initial model, and their private interaction data remains undisclosed to the cloud server.

4.3 Personalized Collaborative Learning Mechanism

When devices obtain their respective initial models, they will relay on training them locally with their newly generated data to make them adapt to user's timely preference. The local training procedure aligns with the learning paradigm of FL, where users train their initial device models locally and share the device model parameters for aggregation. The purpose of model aggregation is to enable collaboration among users, aiming to draw beneficial knowledge from multiple models to enhance the local performance of each user. However, not all model aggregations are advantageous, especially when the preferences of users vary a lot [1, 17]. Because the substantial variance in user preferences can result in significant distribution differences among users' local training data, causing distinctive model update directions during local training. Consequently, straightforwardly aggregating all model parameters cannot yield optimal performance. To enable efficient collaboration among users, we focus on how to aggregate device models that can maximize the local performance of users.

To promote productive collaboration among users, it is imperative to preserve the personalization of models during the model aggregation process. Therefore, we propose to partially aggregate device model parameters between the users with similar preferences. These users usually have similar local data distributions, therefore, aggregating their updated model parameters can effectively achieve desired collaborative benefits akin to augmenting the model training data volume. At the same time, this approach also preserves model personalization, since it mitigates the adverse impact of outlier model updates resulting from users with substantial data distribution disparities. To achieve this, we propose to perform personalized device model aggregation based on the choice of user's initial models. Given that the learned model selector network chooses initial models for users according to their preferences, the users that select the same initial model tend to have similar preferences. Therefore, we propose to aggregate the device model parameters among the users who download the same initial model.

The personalized collaborative learning mechanism of RelayRec is illustrated in Algorithm 3. First, the learned model selector at the cloud server chooses an initial device model for each user based on their new interaction data. Each user downloads the corresponding initial device model from the cloud server and starts the collaborative learning process. In each round, a subset of users are randomly selected to participate in learning. They update the initial model locally based on the FedProx algorithm [27] with their local data and upload the updated device model parameters to the cloud server for aggregation. The cloud server aggregates the model parameters of the users that download identical initial models with the FedAvg [30] algorithm, and the aggregated model parameters are returned to corresponding users to update their local device model. By repeating the iterative steps described above, users' device models are enhanced iteratively for progressively improved local performance.

5 EVALUATION

5.1 Methodology

5.1.1 Experimental Setup.

Base Datasets. We conduct experiments with three publicly accessible datasets and an industrial dataset from Meituan. Public datasets include MovieLens[1], Frappe[2], and KKBox[3], which are widely used in previous CTR prediction studies. To guarantee standard data partitions and preprocessing steps, we employ the preprocessed data[4] accessible from the BARS benchmark [49], which creates an open benchmarking standard for various recommendation tasks. The details of these datasets are introduced as follows.

- **MovieLens:** It contains users' tagging records on movies. The learning task is personalized tag recommendation, where each tagging record (user ID, movie ID, and tag) is converted to a feature vector as input, and the target value represents whether the user has assigned a particular tag to the movie.
- **Frappe:** It consists of app usage logs recorded from users in various contexts (e.g., daytime, location). Each log (user ID, app ID, context features) is converted to a feature vector as input, and the target value indicates whether the user has used the app under the context.
- **KKBox:** It is composed of users' listening history of songs. The task is to predict the chances of a user listening to a song repetitively, where each record (e.g., user id, song id) is converted to a feature vector as input, and the target denotes whether there are recurring listening events triggered within a month.
- **Meituan:** It is constructed from 8-day click logs of Meituan users, spanning two distinct periods: from December 17, 2022, to December 20, 2022, and from February 17, 2023, to February 20, 2023. It encompasses a wide range of information, including user attributes (e.g., user ID, age, gender), item attributes (e.g., item ID, category, price), and context information (e.g., time, location, device type), etc. Each record is converted to a feature vector as input, and the target denotes whether the user has clicked the item.

Dataset Settings. For the three publicly available datasets, we implemented a simulated data regulation scenario by randomly dividing the entire training dataset into two subsets: cloud data and device data. To control the proportion of data collected in the cloud, we introduced a hyperparameter denoted as α. Additionally, we performed a shuffling operation on the user's data field of user ID to simulate changes in user preferences. Specifically, for the Meituan dataset, we designate the data collected from December 17, 2022, to December 20, 2022, as the cloud data gathered prior to the data regulations. Subsequently, we considered the data generated from February 17, 2023, to February 20, 2023, as the data produced after the regulations came into effect and are decentralized on the user devices. This scenario allowed us to evaluate RelayRec's performance under the impact of data regulations and changing user preferences.

Models. We select two lightweight CTR prediction models: LR and DNN as device models, and select the advanced xDeepFM model as the server model. We briefly describe these models as follows.

- **LR:** Logistic regression (LR) is a simple model for CTR prediction [36], which sums up raw features linearly.
- **DNN:** It is a deep model proposed in [5], which utilizes a fully-connected network following the concatenation of feature embeddings for CTR prediction.
- **xDeepFM:** It integrates a compressed interaction network (CIN) with a DNN to capture high-order feature interactions for CTR prediction.

5.1.2 Baselines. We compare RelayRec with the following baselines. The main distinctions between them and RelayRec can be summarized in two aspects: 1) how to utilize the cloud data to learn

[1]https://grouplens.org/datasets/movielens
[2]https://www.baltrunas.info/context-aware
[3]https://www.kaggle.com/c/kkbox-musicrecommendation-challenge
[4]https://github.com/openbenchmark/BARS/tree/main/datasets

Table 2: Performance comparison between RelayRec and baselines.

Base Model	Initial Model	Dataset Method	MovieLens		Frappe		KKBox		Meituan	
			AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
LR	Normal-	Standalone	0.843	0.639	0.858	0.644	0.650	0.647	0.720	0.541
		FedAvg	0.849	0.629	0.752	0.581	0.648	0.647	0.723	0.493
		FedProx	0.882	0.579	0.866	0.538	0.649	0.647	0.718	0.501
		Clustering	0.840	0.590	0.777	0.580	0.611	0.668	0.722	0.493
	Meta-	Standalone	0.812	0.611	0.842	0.619	0.655	0.647	0.716	0.538
		FedAvg	0.828	0.612	0.775	0.576	0.657	**0.641**	0.721	0.497
		FedProx	0.862	0.568	0.867	0.531	0.653	0.644	0.719	0.497
		Clustering	0.829	0.581	0.805	0.575	0.657	0.642	0.720	0.499
	RelayRec		**0.885**	**0.532**	**0.877**	**0.530**	**0.661**	0.642	**0.725**	**0.491**
DNN	Normal-	Standalone	0.808	0.492	0.835	0.549	0.625	0.694	0.673	1.416
		FedAvg	0.793	0.483	0.825	0.544	0.619	0.662	0.663	0.619
		FedProx	0.876	0.464	0.817	0.530	0.627	0.660	0.680	0.559
		Clustering	0.847	0.450	0.822	0.520	**0.649**	**0.647**	0.661	0.566
	Meta-	Standalone	0.816	0.492	0.821	0.552	0.614	0.687	0.584	0.734
		FedAvg	0.840	0.493	0.816	0.486	0.620	0.661	0.662	0.548
		FedProx	0.855	0.498	0.819	0.575	0.640	0.654	0.685	0.557
		Clustering	0.843	0.450	0.820	0.548	0.626	0.666	0.677	**0.548**
	RelayRec		**0.883**	**0.421**	**0.878**	**0.422**	0.634	0.664	**0.733**	0.583

initial device models for users; and 2) how to perform collaborative learning among users with on-device data after regulations.

There are two common approaches that can be used to learn initial device models for users: one is training a device model with the cloud data directly, and the other is learning a meta-model applying meta-learning. We denote these two approaches as **Normal-Init.-Model** and **Meta-Init.-Model** respectively. For relay learning of initial models on devices, the following approaches can be applied:

- **Standalone:** Each user finetunes the initial model on device with their local data without collaboration with any other users.
- **FedAvg:** Collaborative learning is performed among users with the FedAvg algorithm [30], which is a basic method in FL.
- **FedProx:** It is a popular federated optimization algorithm [27], which introduces a proximal term to the local training objective of devices to enhance the FedAvg algorithm under non-IID data.
- **Clustering:** It is a personalized FL method [12, 33, 37], which clusters users into groups based on their local data distributions and learns a personalized shared model within each group.

5.1.3 Parameter Settings. We set $\alpha = 5\%$ by default to set up the experimental scenarios for performance evaluation. For the baseline approaches and RelayRec, we adopt the Adam optimizer with step size $\eta = 0.001$. We set the local epochs $\tau = 5$, the sampled user number per round $N = 25$. The default settings of RelayRec are user cluster number $K = 2$, step size $\eta_1 = 0.01, \eta_2 = 0.01, \eta_3 = 0.005$, item cluster number $C = 2$.

5.1.4 Evaluation Metrics. As with many previous studies, we adopt two widely used metric, i.e., logloss and AUC, to report the evaluation results.

- **AUC** quantifies the probability that a randomly selected positive sample is ranked higher than a randomly selected negative sample. A higher AUC value indicates better CTR prediction performance.
- **Logloss** is also referred to as the binary cross-entropy loss, serves as a metric for quantifying the loss in a binary classification scenario. A lower Logloss value indicates better CTR prediction performance.

5.2 Performance Comparison

We first conduct a performance comparison of RelayRec with all of the baselines on four datasets using two CTR prediction models. The results are reported in Table 2. It can be observed that RelayRec consistently outperforms the baselines in most cases. Particularly, it demonstrates the most substantial performance gains on the Frappe dataset, showing an average of 7.2% AUC improvement and 14.9% LogLoss reduction. Moreover, RelayRec also notably enhances performance on the Meituan industrial dataset, achieving approximately 5.9% improvement in AUC and 5.4% reduction in LogLoss. Overall, RelayRec boosts AUC by 5% and reduces LogLoss by 8.1% on average. These performance improvements demonstrate the superiority of RelayRec in privacy-preserving CTR prediction. In addition, there are some noteworthy observations. First, on the KKBox dataset, the DNN model exhibits superior performance when used with the Clustering approach and a normal-initial model. We delve into this phenomenon and identified the reason behind it. In KKBox, a large number of users have substantial local data, which promotes the local training performance of devices and dilutes the benefits of meta-learning. In contrast, RelayRec is more competitive in other datasets where each user has relatively small-scale

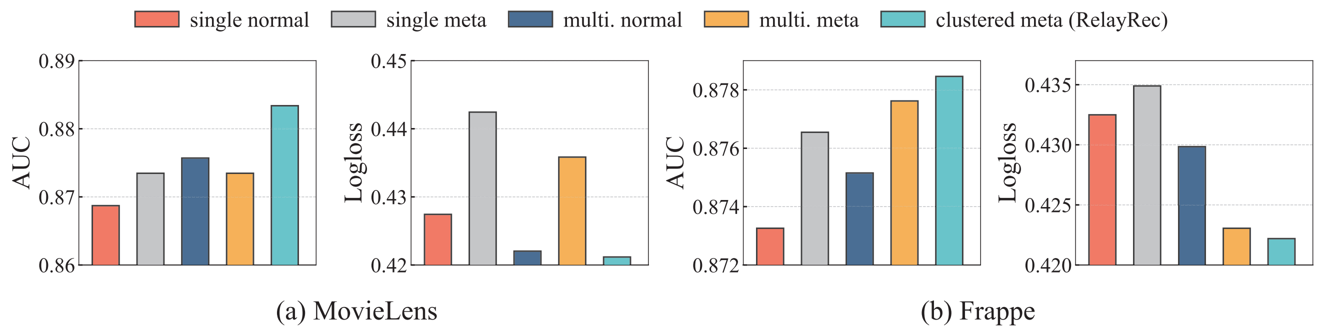

Figure 4: Performance of RelayRec with different initial model learning approaches.

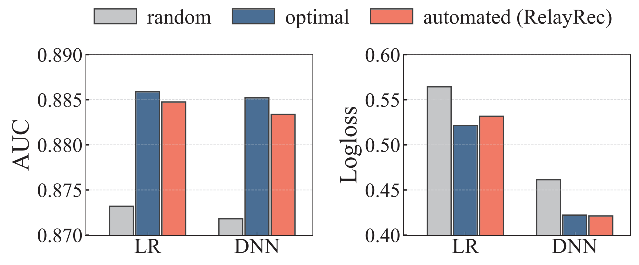

Figure 5: Performance of RelayRec with different initial model selection approaches.

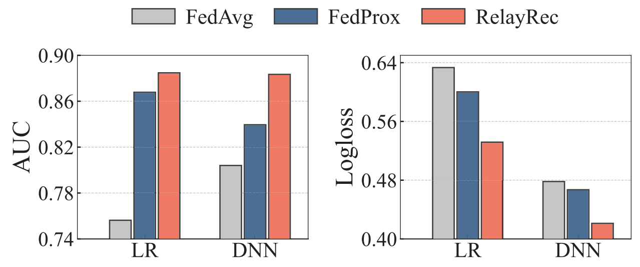

Figure 6: Performance of RelayRec with different collaborative learning approaches.

local data. Furthermore, we notice that the FedProx method is particularly competitive among the baseline approaches. Since the FedProx method is proposed to solve the non-IID problem in FL, its performance advantage emphasizes the importance of efficient collaborative learning for on-device prediction performance. Nevertheless, in addition to the collaborative learning efficiency, the initial model for devices also plays a vital role. RelayRec takes both of these factors into account and therefore enables outperformed performance.

5.3 Breakdown Evaluation

This section conducts a breakdown evaluation of RelayRec by assessing the individual performance contributions of its key components: the clustered meta-learning mechanism, the automated model selector mechanism, and the personalized collaborative learning mechanism.

Effectiveness of the clustered meta-learning mechanism. To evaluate the effectiveness of the proposed clustered meta-learning mechanism, we conduct a series of comparisons by substituting the clustered meta-learning mechanism with alternative approaches within the RelayRec framework. The evaluations focus on diverse methods for learning initial device models and their subsequent effects on RelayRec's overall performance. The evaluated alternative approaches included: 1) *single normal*: learning a single device model for all users with the normal training paradigm; 2) *single meta*: learning a single meta-model for all users with meta-learning; 3) *mutiple normal*: clustering users with the same method as RelayRec and learning a normal model within each cluster to derive multiple initial models; 4) *multiple meta*: clustering users with the same method as RelayRec and learning a meta-model within each cluster to derive multiple meta-models. We select the MovieLens and Frappe datasets along with the DNN model for our study, and

Fig. 4 illustrates the performance of RelayRec with different initial model learning approaches. It can be observed that the proposed clustered meta-learning mechanism enables the optimal performance of RelayRec on both datasets, which demonstrates its superior performance against other initial model learning methods. Besides, we also notice that user clustering alone does not bring desirable performance gains. This is because while clustering enables personalization of initial models, it also reduces the amount of training data for each initial model, thereby impacting the learning performance. This is also one of the reasons why we further finetune the learned meta-initial-models with the total cloud data in the proposed clustered meta-learning mechanism.

Effectiveness of the automated model selector mechanism. The automated model selector mechanism in RelayRec is proposed to learn a model selector network to choose initial device models for users. To demonstrate its effectiveness, we replace the model selection mechanism in RelayRec with alternative methods and then evaluate the performance of RelayRec. Here, we focus on two alternative model selection approaches: 1) *random*: users' initial device models are randomly selected; 2) *optimal*: this approach evaluates the performance of each initial device model on the local data of each user and the model yielding the highest performance is selected as the initial model. The evaluations are performed on the MovieLens dataset and Fig. 5 shows the performance of RelayRec with different model selection mechanisms. From the results, we can observe that the performance of the proposed automated model selector outperforms the random method significantly and can even be competitive with the optimal method. The results demonstrate that the automated model selector mechanism of RelayRec can effectively choose advantageous initial models for users, promoting a favorable relay learning performance.

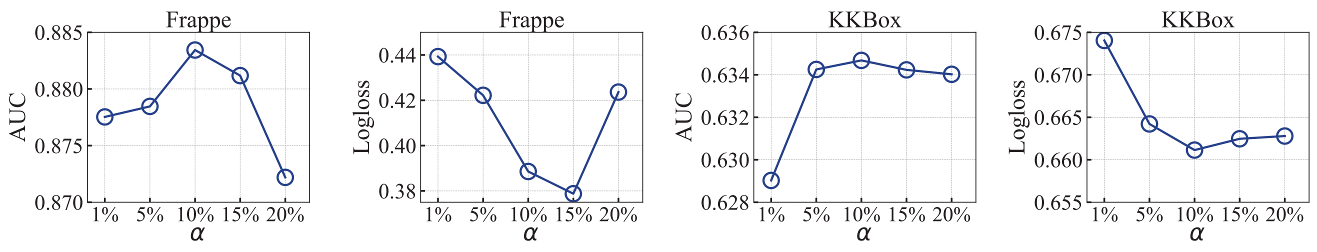

Figure 7: DNN model learning performance with RelayRec under varying cloud data ratio α.

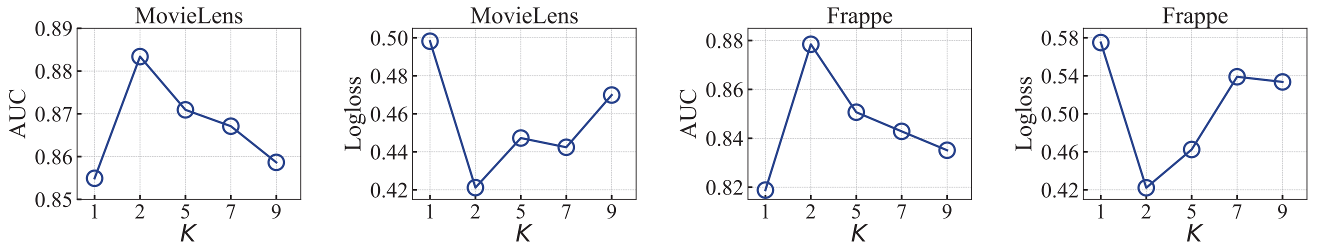

Figure 8: DNN model learning performance with RelayRec under varying user cluster number K.

Effectiveness of the personalized collaborative learning mechanism. To evaluate the efficacy of RelayRec's personalized collaborative learning mechanism, a performance comparison is conducted with existing collaborative learning approaches. Here, we compare with two popular methods in FL, i.e., FedAvg and FedProx[5]. To guarantee fair comparison, we also employ RelayRec's clustered meta-learning method to learn initial device models and RelayRec's automated model selector to choose initial models for the baseline collaborative learning approaches. The evaluations are conducted on the MovieLens dataset, and the performance comparison results of RelayRec with different collaborative learning approaches are shown in Fig. 6. It can be observed that the proposed personalized collaborative learning mechanism performs better than the baseline collaborative learning approaches. Besides, we notice that the FedProx method enables a considerable performance improvement compared to the FedAvg method. We combine RelayRec with the FedProx method and then observe further performance enhancements in RelayRec. The experimental results demonstrate the effectiveness and compatibility of RelayRec's personalized collaborative learning approach.

5.4 Sensitivity Analysis

This section evaluates the performance of RelayRec under various parameter settings. Here, we focus on two important parameters, i.e., cloud data ratio α and the user cluster number K.

First, we investigate the sensitivity of RelayRec to the ratio of cloud data. To this end, we vary the value of α that controls the ratio of cloud data from 1% to 20% and show the corresponding performance of RelayRec in Fig. 7. We observe that the performance of RelayRec on the two datasets initially improves with the increase in cloud data and then decreases. Actually, as the ratio of the cloud data increases, RelayRec is consistently balancing between the benefits of increased cloud data and the disadvantages of reduced on-device data. Within RelayRec, cloud data serves two purposes: one is for training the initial device models for users, and the other is for training the model selector. The increase in cloud data results in better initial device models and enhanced performance of the model selector. In contrast, on-device data is user's newly generated data, indicating their timely preferences. Thus, the increase in on-device data contributes to adapting device models to users' varying preferences. Therefore, both the cloud data and on-device data play a vital role in the performance of RelayRec.

Then we explore the impact of the user cluster number settings on the performance of RelayRec. We vary the setting of user cluster number K from 1 to 9 and show the perference of RelayRec in Fig. 8. From the results, we can observe that the performance of RelayRec first improves and then declines with the increase of the cluster number. It happens since clustering enables preference-specific and improved initial models, which is beneficial for users to acquire a well-initialized device model. However, on the other hand, an increase in the number of clusters also amplifies the training difficulty of the model selector, which poses challenges to find suitable initial models for users. The results suggest that the cluster number has a significant impact on the performance of RelayRec, and the trade-off between improved initial models and selector network effectiveness must be carefully balanced to achieve the best overall performance for RelayRec.

6 DISCUSSION

This section discusses the limitations of RelayRec and explores potential approaches to address them.

Experimental Settings. In this paper, we focus on a practical scenario where users' preferences vary over time. However, most publicly available CTR prediction datasets involve a relatively short time spans, and users' preferences have not experienced significant changes. Therefore, we manually create experimental scenarios where user preferences undergo changes to demonstrate the effectiveness of RelayRec. Nevertheless, we have demonstrated the efficacy of RelayRec on the Meituan industrial dataset. In our future work, we will further extend the Meituan dataset to cover a longer

[5]The clustering method was not included in the comparison since RelayRec's personalized collaborative learning approach builds upon the principles of clustering.

time span, and show the effectiveness and economic benefits of RelayRec in practice.

Time-Varying User Preference. In the evaluations, we employ an experimental setting where the data distribution of users after data regulations is different from before to simulate the preference changes of users. In practice, since user data is generated continuously, the preferences of users may change frequently. Therefore, models can continuously update on devices with newly generated data to adapt to users' evolving preferences. Additionally, when a user's preference changes, the model selector will reselect an initial model tailored to the updated preference. Subsequently, users download the corresponding initial models from the cloud server and re-clustered performing collaborative learning. In our future work, we will incorporate detection algorithms to RelayRec to enable automatic identification of the changes in user preferences.

Limitations and Future Enhancements. In RelayRec, we propose to group users and learn initial device models according to user preferences. However, how to characterize user preferences remains an open problem. The solution in RelayRec derives user embeddings for user clustering by learning an advanced CTR prediciton model with the cloud data. Although this solution may not be optimal, it possesses other potentials. For example, when learning initial device models at the cloud server, the pre-trained advanced server model can be employed as a teacher model, facilitating knowledge transfer to initial device models for advanced performance. In the future work, we will explore more tailored methods for characterizing user preferences in RelayRec, aiming to enhance the overall performance of the system.

7 RELATED WORK

7.1 CTR Prediction

The advent of deep learning has significantly contributed to enhancing the performance of CTR prediciton, and deep CTR models have found widespread application across various industrial platforms, e.g., Wide&Deep [4] of Google, DeepFM [13] of Huawei, DIN [48] of Alibaba, and xDeepFM [28] of Microsoft. These deep learning-based CTR prediction methods usually rely on the user behavior data collected by the platform to model user interest. Generally, a historical set of instances with pre-defined size, e.g., the previous 7-day logs, are retrieved as the training samples after performing feature engineering. Then the learned CTR prediction model will be deployed in practice for inference. Since the deployment environment is usually dynamic, the model requires regular re-training to capture time-changing user behaviors from newly collected interaction logs. However, the pipeline of training and deployment commonly takes a long time, which undermines the freshness and performance of the deployed models. To address this issue, previous works proposed continual learning frameworks for CTR prediction [3, 16, 20]. For example, Katsileros *et al.* in [20] employed a teacher-student incremental learning paradigm, where the teacher acts as an implicit regularizer, enabling the student to maintain previously learned knowledge. Hu *et al.* in [16] introduced a memory replay module to mitigate catastrophic forgetting of previous knowledge and promote positive knowledge transfer. Although these studies also considered the dynamic nature of user behaviors, they assume that user data can be collected centrally without taking privacy into consideration. Moreover, the goal of RelayRec is to enable the model to adapt more quickly and effectively to users' new data, rather than preventing the model from forgetting previously acquired knowledge, which differentiates RelayRec from these approaches. Besides, the relay learning paradigm of RelayRec directly updates user models on devices, which also reduces the cost associated with frequent model training and deployment for delivering new model versions in real-world scenarios.

7.2 Federated Learning

In RelayRec, the relay learning on user devices is based on FL. Due to the increasing emphasis on privacy protection in recommendation systems and advertising, a bunch of research proposed FL-based solutions. FedFast [31] introduced an active user sampling and model aggregation approach for FL to expedite the convergence speed of a matrix factorization model. SFSL [32] introduced submodel training on devices to enhance the efficiency and privacy of federated recommendation systems. PEPPER [1] allows users to asynchronously train personalized models based on gossip learning principles. PREFER [14] proposed to share user-independent parameters across users in federated Point-of-Interest recommendation systems. Wu et al. in [43] proposed a federated native ad CTR prediction method to learn user-interest representations from cross-platform user behaviors in a privacy-preserving way. RelayRec differs from these works in several aspects. First, RelayRec aims to improve the local performance of users, and hence it learns personalized models that can adapt to user's local data instead of learning a shared global model for all users. Second, it emphasizes the benefits of pre-collected cloud data and fully utilizes them for efficient relay learning, which is ignored by previous personalized FL works. Besides, RelayRec is also different from previous FL-based solutions for recommendation systems or advertising, as it proposes to fully exploit the data collected before data regulations to provide well-initialized models for devices via clustered meta-learning, and efficiently utilize the private data decentralized on devices to relay on training models via personalized collaborative learning.

8 CONCLUSION

In this paper, we propose RelayRec, a cloud-device relay learning framework for privacy-preserving CTR prediction. RelayRec takes benefits of the cloud data collected before data regulations and the private user data decentralized on device after data regulations to improve the local CTR prediction performance of users. At the core of RelayRec resides three key components: 1) a clustered meta-learning mechanism, which divides cloud data into clusters based on user's preference and trains multiple preference-wise device models for users as their initial models; 2) an automated model selector mechanism that can intelligently choose a beneficial initial device model for each user based on their local data; and 3) a personalized collaborative learning mechanism, which enables users with similar preferences collaboratively learn a personalized device model utilizing their local private data in a privacy-preserving way. Extensive evaluations on various real-world datasets demonstrate the superiority of RelayRec compared to state-of-the-art approaches.

9 ACKNOWLEDGEMENT

This research was supported in part by the National Key R&D Program of China under Grant No. 2022YFF0604502, the National Natural Science Foundation of China under Grant No. 62122095, 62341201 and 62072472, and by a grant from the Guoqiang Institute, Tsinghua University.

REFERENCES

[1] Yacine Belal, Aurélien Bellet, Sonia Ben Mokhtar, and Vlad Nitu. 2022. PEPPER: Empowering user-centric recommender systems over gossip learning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 3 (2022), 1–27.

[2] Ilai Bistritz, Ariana Mann, and Nicholas Bambos. 2020. Distributed distillation for on-device learning. In *NeurIPS*. 22593–22604.

[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 *ACM SIGIR*. 2692–2697.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7–10.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *ACM RecSys*. 191–198.

[6] Yongheng Deng, Weining Chen, Ju Ren, Feng Lyu, Yang Liu, Yunxin Liu, and Yaoxue Zhang. 2022. TailorFL: Dual-Personalized Federated Learning under System and Data Heterogeneity. In *ACM SenSys*. 592–606.

[7] Yongheng Deng, Feng Lyu, Ju Ren, Huaqing Wu, Yuezhi Zhou, Yaoxue Zhang, and Xuemin Shen. 2021. AUCTION: Automated and Quality-Aware Client Selection Framework for Efficient Federated Learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1996–2009.

[8] Yongheng Deng, Ju Ren, Cheng Tang, Feng Lyu, Yang Liu, and Yaoxue Zhang. 2023. A Hierarchical Knowledge Transfer Framework for Heterogeneous Federated Learning. In *IEEE INFOCOM*.

[9] Manqing Dong, Feng Yuan, Lina Yao, Xiwei Xu, and Liming Zhu. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In *ACM SIGKDD*. 688–697.

[10] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. In *NeurIPS*. 3557–3568.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*. 1126–1135.

[12] Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. 2020. An efficient framework for clustered federated learning. *NeurIPS* 33 (2020), 19586–19597.

[13] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *IJCAI*. 1725–1731.

[14] Yeting Guo, Fang Liu, Zhiping Cai, Hui Zeng, Li Chen, Tongqing Zhou, and Nong Xiao. 2021. PREFER: Point-of-interest REcommendation with efficiency and privacy-preservation via Federated Edge leaRning. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 1 (2021), 1–25.

[15] Jialiang Han, Yun Ma, Qiaozhu Mei, and Xuanzhe Liu. 2021. DeepRec: On-device deep learning for privacy-preserving sequential recommendation in mobile commerce. In *WWW*. 900–911.

[16] Ke Hu, Yi Qi, Jianqiang Huang, Jia Cheng, and Jun Lei. 2022. Continual Learning for CTR Prediction: A Hybrid Approach. *arXiv preprint arXiv:2201.06886* (2022).

[17] Mubashir Imran, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Alexander Zhou, and Kai Zheng. 2023. ReFRS: Resource-efficient federated recommender system for dynamic and diversified user preferences. *ACM Transactions on Information Systems* 41, 3 (2023), 1–30.

[18] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *ACM RecSys*. 43–50.

[19] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *ICML*. 5132–5143.

[20] Petros Katsileros, Nikiforos Mandilaras, Dimitrios Mallis, Vassilis Pitsikalis, Stavros Theodorakis, and Gil Chamiel. 2022. An Incremental Learning framework for Large-scale CTR Prediction. In *ACM RecSys*. 490–493.

[21] Viraj Kulkarni, Milind Kulkarni, and Aniruddha Pant. 2020. Survey of Personalization Techniques for Federated Learning. *arXiv preprint arXiv:2003.08673* (2020).

[22] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *OSDI*. 19–35.

[23] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In *ACM SIGKDD*. 1073–1082.

[24] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *ACM MobiCom*. 420–437.

[25] Ang Li, Jingwei Sun, Xiao Zeng, Mi Zhang, Hai Li, and Yiran Chen. 2021. FedMask: Joint Computation and Communication-Efficient Personalized Federated Learning via Heterogeneous Masking. In *ACM SenSys*. 42–55.

[26] Daliang Li and Junpu Wang. 2019. FedMD: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).

[27] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. In *MLSys*. 429–450.

[28] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *ACM SIGKDD*. 1754–1763.

[29] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *WWW*. 689–698.

[30] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[31] Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. FedFast: Going beyond average for faster training of federated recommender systems. In *ACM SIGKDD*. 1234–1242.

[32] Chaoyue Niu, Fan Wu, Shaojie Tang, Lifeng Hua, Rongfei Jia, Chengfei Lv, Zhihua Wu, and Guihai Chen. 2020. Billion-scale federated learning on mobile clients: A submodel design with tunable privacy. In *ACM MobiCom*. 1–14.

[33] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. ClusterFL: a similarity-aware federated learning system for human activity recognition. In *MobiSys*. 54–66.

[34] Feiyang Pan, Shuokai Li, Xiang Ao, Pingzhong Tang, and Qing He. 2019. Warm up cold-start advertisements: Improving ctr predictions via learning to learn id embeddings. In *ACM SIGIR*. 695–704.

[35] Haoyu Pang, Fausto Giunchiglia, Ximing Li, Renchu Guan, and Xiaoyue Feng. 2022. PNMTA: A pretrained network modulation and task adaptation approach for user cold-start recommendation. In *WWW*. 348–359.

[36] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. 521–530.

[37] Yichen Ruan and Carlee Joe-Wong. 2021. FedSoft: Soft Clustered Federated Learning with Proximal Local Updating. *arXiv preprint arXiv:2112.06053* (2021), 10.

[38] Canh T Dinh, Nguyen Tran, and Josh Nguyen. 2020. Personalized federated learning with moreau envelopes. In *NeurIPS*. 21394–21405.

[39] Ricardo Vilalta and Youssef Drissi. 2002. A perspective view and survey of meta-learning. *Artificial intelligence review* 18 (2002), 77–95.

[40] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM*. 1698–1707.

[41] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *NeurIPS*. 7611–7623.

[42] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[43] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2022. FedCTR: Federated native ad CTR prediction with cross-platform user behavior data. *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, 4 (2022), 1–19.

[44] Mengwei Xu, Feng Qian, Qiaozhu Mei, Kang Huang, and Xuanzhe Liu. 2018. Deeptype: On-device deep learning for input personalization service with minimal privacy concern. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–26.

[45] Jieyu Yang, Zhaoxin Huan, Yong He, Ke Ding, Liang Zhang, Xiaolu Zhang, Jun Zhou, and Linjian Mo. 2022. Task Similarity Aware Meta Learning for Cold-Start Recommendation. In *ACM CIKM*. 4630–4634.

[46] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. 2019. Hierarchically structured meta-learning. In *ICML*. 7045–7054.

[47] Jiangchao Yao, Feng Wang, Kunyang Jia, Bo Han, Jingren Zhou, and Hongxia Yang. 2021. Device-cloud collaborative learning for recommendation. In *ACM SIGKDD*. 3865–3874.

[48] Guorui Zhou, Xiaoqiang Zhu, Chenru 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 *ACM SIGKDD*. 1059–1068.

[49] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. Bars: Towards open benchmarking for recommender systems. In *ACM SIGIR*. 2912–2923.