FedCTR: Federated Native Ad CTR Prediction with Cross Platform User Behavior Data

CHUHAN WU, Department of Electronic Engineering, Tsinghua University, China
FANGZHAO WU*, Microsoft Research Asia, China
LINGJUAN LYU, Sony AI, Japan
YONGFENG HUANG*, Department of Electronic Engineering, Tsinghua University, China
XING XIE, Microsoft Research Asia, China

Native ad is a popular type of online advertisement which has similar forms with the native content displayed on websites. Native ad click-through rate (CTR) prediction is useful for improving user experience and platform revenue. However, it is challenging due to the lack of explicit user intent, and user behaviors on the platform with native ads may be insufficient to infer users' interest in ads. Fortunately, user behaviors exist on many online platforms which can provide complementary information for user interest mining. Thus, leveraging multi-platform user behaviors is useful for native ad CTR prediction. However, user behaviors are highly privacy-sensitive, and the behavior data on different platforms cannot be directly aggregated due to user privacy concerns and data protection regulations. Existing CTR prediction methods usually require centralized storage of user behavior data for user modeling, which cannot be directly applied to the CTR prediction task with multi-platform user behaviors. In this paper, we propose a federated native ad CTR prediction method named FedCTR, which can learn user interest representations from cross-platform user behaviors in a privacy-preserving way. On each platform a local user model learns user embeddings from the local user behaviors on that platform. The local user embeddings from different platforms are uploaded to a server for aggregation, and the aggregated ones are sent to the ad platform for CTR prediction. Besides, we apply local differential privacy (LDP) and differential privacy (DP) to the local and aggregated user embeddings respectively for better privacy protection. Moreover, we propose a federated framework for collaborative model training with distributed models and user behaviors. Extensive experiments on real-world dataset show that FedCTR can effectively leverage multi-platform user behaviors for native ad CTR prediction in a privacy-preserving manner.

Additional Key Words and Phrases: Native Ad, CTR Prediction, Federated Learning, Privacy-preserving

1 INTRODUCTION

Native ad is a popular form of online advertisements that has similar styles and functions with the native content displayed on online platforms, such as news and video websites [25]. An illustrative example of native ads on the homepage of MSN News[1] is shown in Figure 1. We can see that except for a sign of "Ad", the appearance of the embedded native ads is very similar with the listed news articles. Due to the reduction of ad recognition, native ads can better attract users' attentions, and have gained increasing popularity in many online platforms [40].

*Corresponding Authors
[1]https://www.msn.com/en-us

Authors' addresses: Chuhan Wu, Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China, wuchuhan15@gmail.com; Fangzhao Wu, Microsoft Research Asia, Beijing, China, 100080, wufangzhao@gmail.com; Lingjuan Lyu, Sony AI, 1-7-1 Konan Minato-ku, Tokyo, 108-0075, Japan, lingjuanlvsmile@gmail.com; Yongfeng Huang, Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China, yfhuang@tsinghua.edu.cn; Xing Xie, Microsoft Research Asia, Beijing, China, 100080, xing.xie@microsoft.com.

2157-6904/2022/1-ART1 $15.00
https://doi.org/10.1145/3506715

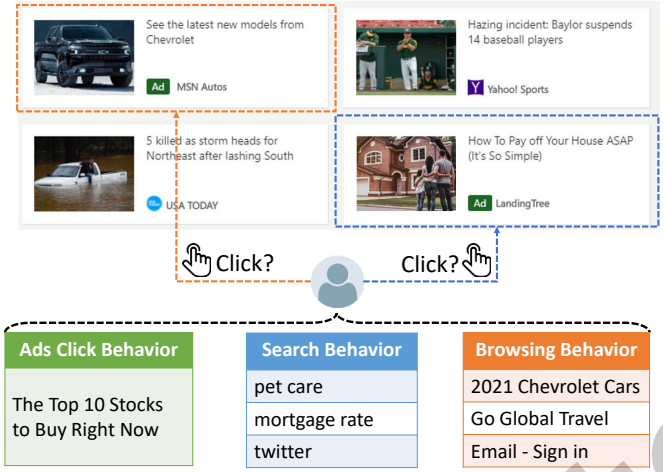


Fig. 1. An illustrative example of native ads and user behaviors on different platforms.

Therefore, accurate click-through rate (CTR) prediction of native ads is an important task for online advertising, which can improve users' experience by recommending their interested ads as well as improve the revenue of online websites by attracting more ad clicks [4].

Although the CTR prediction for search ads and display ads has been widely studied [17, 44], the research on native ads is very limited [1]. Compared with search ads which are displayed based on users' intents inferred from their search queries, there is no explicit user intent for native ads, making it more difficult to predict the click probability. By contrast, display ads are usually presented on e-commerce platforms, and their CTR prediction can take the advantage of various user behavior records such as browsing, preferring and purchasing for interest modeling. However, users' behaviors on the platform where native ads are displayed usually cannot provide sufficient information for inferring their interest in ads, such as the news reading behaviors at online news websites.

Fortunately, users' online behaviors exist in many different platforms, and they can provide various clues to infer user interest in different aspects. For example, in Figure 1, the search behaviors on search engine platform (e.g., "mortgage rate") indicate that this user may have interest in buying a new house or applying for housing loan, and she may click the native ad with title "How to Pay Off Your House ASAP". In addition, from her webpage browsing behaviors we can infer that this user may be interested in cars since she browsed a webpage with title "2021 Chevrolet Cars", and it is appropriate to display the native ad "See the latest new models from Chevrolet" to her. Thus, incorporating user behaviors on multiple platforms is useful for modeling user interest more accurately and can benefit native ad CTR prediction, which has been validated by existing studies [1]. For example, An et al. [1] found that combining users' searching behaviors and webpage browsing behaviors can achieve better performance on native ad CTR prediction than using single kind of user behaviors. However, online user behaviors such as the queries they searched and the webpages they browsed are highly privacy-sensitive. Thus, the behavior data on different platforms cannot be directly aggregated into a single server or exchanged among different platforms due to users' privacy concerns and the user data protection regulations like GDPR[2]. Most existing CTR prediction methods rely on centralized storage of user behavior data, making them difficult to be applied to the native ad CTR prediction task with multi-platform user behaviors.

[2]https://gdpr-info.eu

In this paper, we propose a federated native ad CTR prediction method named FedCTR, which can incorporate the user behavior data on different platforms to model user interest in a privacy-preserving way without storing them centrally. Specifically, we learn unified user representations from different platforms in a federated way. On each platform that participates in user representation learning, a local user model is used to learn the local user embeddings from the user behavior logs on that platform. The learning of user embeddings on different platforms is coordinated by a user server, and these local embeddings are uploaded to this server for aggregation. The aggregated unified user embeddings are further sent to the ad platform for CTR prediction. Thus, the FedCTR method can exploit the user information on different platforms for native ad CTR prediction. Since the raw user behavior logs never leave the local platform and only user embeddings are uploaded, user privacy can be protected to a certain extent in the FedCTR method. In addition, we apply local differential privacy (LDP) [34] and differential privacy (DP) [7] techniques to the local and aggregated user embeddings respectively before sending them, which can achieve better privacy protection at a small cost of CTR prediction performance. Since in FedCTR the user behavior data and model parameters are located on different platforms, it is a non-trivial task to train the model for FedCTR without exchanging the privacy-sensitive user behavior data across different platforms. Thus, we propose a privacy-preserving framework based on federated learning to address this issue. In our framework, only user embeddings and intermediate gradients are communicated among different platforms, thus the risk of user privacy leakage can be effectively reduced at the model training stage. Extensive experiments are conducted on a real-world dataset collected from the user logs on the native ads in the MSN News website[3]. The results validate that our approach can improve the performance of native ad CTR prediction via exploiting the multi-platform user behaviors and meanwhile effectively protect user privacy.

The main contributions of this work include:

- We propose a privacy-preserving native ad CTR prediction method named FedCTR which can exploit multi-platform user behaviors for user interest modeling without centralized storage of them.
- We design a federated model training framework to train models for FedCTR where user behavior data and model parameters are distributed on different platforms.
- We conduct extensive experiments on a real-world dataset to verify the effectiveness of the proposed method in both CTR prediction and privacy protection.

2 RELATED WORK

2.1 CTR Prediction

Native ad is a special kind of display ads which has similar form with the native content displayed in online websites [25]. CTR prediction for display ads has been extensively studied [5, 10, 19, 21, 23, 37, 44, 45]. Different from search ads where the search query triggering ad impressions can provide clear user intent, in display ads there is no explicit user intent [45]. Thus, it is very important for display ad CTR prediction to model user interest from users' historical behaviors. Many existing CTR prediction methods rely on handcrafted features to represent users and ads, and they focus on capturing the interactions between these features to estimate the relevance between users and ads [3, 5, 10, 15, 20, 28, 37, 45]. For example, Cheng et al. [5] proposed a Wide&Deep model that integrates a wide linear channel with cross-product and a deep neural network channel to capture feature interactions for CTR prediction. They represented users' interest with their demographic features, device features and the features extracted from their historical impressions. Guo et al. [10] proposed a DeepFM model that combines factorization machines and deep neural networks to model feature interactions. They used ID and category features to represent ads, and used the feature collections of historical clicked items to represent users. However, the design of handcrafted features used in these methods usually requires massive domain knowledge, and handcrafted features may not be optimal in modeling user interest. There are also several methods for display

[3]https://www.msn.com/en-us

ads CTR prediction that use deep learning techniques to learn user interest representations from their behaviors on e-commerce platforms [9, 18, 31]. For example, Zhou et al. [45] proposed a deep interest network (DIN) that learns representations of users from the items they have interacted with on the e-commerce platform based on the relatedness between those items and the candidate ads. In [44], an improved version of DIN named deep interest evolution network (DIEN) was proposed, which models users from their historical behaviors on items via a GRU network with attentional update gate. These deep learning based CTR prediction methods for display ads on e-commerce platform usually rely on the user behaviors on the same platform to model user interest. However, besides the e-commerce platform, native ads are widely displayed on many other online platforms such as news websites [1]. The user behaviors on these platforms may have insufficient clues to infer user interest in ads. Thus, these CTR prediction methods designed for display ads on e-commerce platform may be not optimal for native ads.

There are only a few studies for native ad CTR prediction [1, 29]. For example, An et al. [1] proposed to model user interest for native ad CTR prediction from their search queries and browsed webpages. These studies found that incorporating multi-platform user behaviors can model user interest more accurately than using single-platform user behaviors. These methods usually require centralized storage of multi-platform user behaviors. However, user behavior data is highly privacy-sensitive, and cannot be directly aggregated across platforms due to privacy concerns and user data protection regulations like GDPR. Different from these methods, our proposed FedCTR method can exploit user behaviors on different platforms for native ad CTR prediction via federated learning, which can eliminate the centralized storage of user behavior data and achieve better user privacy protection.

2.2 Federated Learning

The learning of user representation in our proposed FedCTR method with multi-platform user behavior data is based on federated learning [26]. Federated learning is a recently proposed machine learning technique, which can learn a shared model from the private data of massive users in a privacy-preserving way [8, 11, 22, 27, 42]. Instead of directly uploading the private user data to a central server for model training, in federated learning the user data is locally stored on different user devices such as smartphones and personal computers. Each user device has a copy of the local model and computes the model updates based on local user interactions. The model updates from a large number of users are uploaded to the server and aggregated into a single one for global model update [26]. Then the new global model is delivered to user devices for local model update, and this process iterates for multiple rounds. Since the model updates contain much less information than the raw user data, federated learning can provide an effective way to exploit the private data of different users and protect their privacy at the same time [26]. Based on the idea of federated learning, Jiang et al. [14] proposed a federated topic modeling approach to train topic models from the corpus owned by different parties. In these federated learning methods, the samples for model training are distributed on different clients, and each client shares the same feature space, aka horizontal federated learning [42]. In horizontal federated learning, the models are usually fully or partially shared across different clients, and the training sample sets on different clients are usually different. The server and clients communicate the updates of the local models. However, different from horizontal federated learning, in the task of native ad CTR prediction with multi-platform user behaviors, the user behavior data of each sample is distributed on different platforms. These platforms may contain the same sample but they can only see part of the user feature space, aka vertical federated learning [42]. In vertical federated learning, the models on different clients are usually independent, and the feature sets of each sample are kept by different clients in a decentralized manner. Since the features on different clients cannot be directly aggregated, the clients need to exchange the intermediate results inferred by local models as well as their gradients to synthesize the final prediction results and train models, respectively. Note that different from horizontal federated learning, in

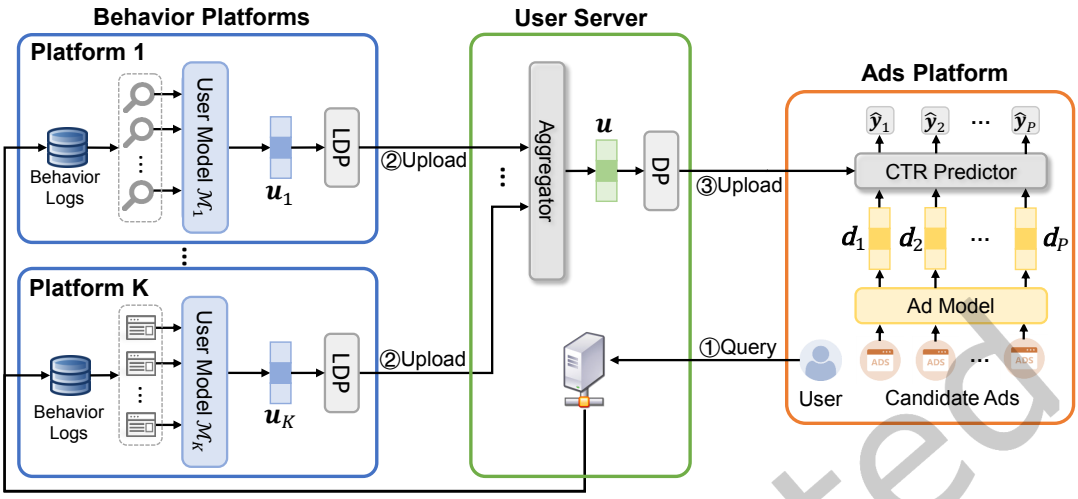


Fig. 2. The architecture of the *FedCTR* method.

vertical federated learning the clients are usually service providing platforms that store certain kinds of data of many users rather than individual user devices. Thus, the problem studied in this work is quite different from the existing federated learning methods. To our best knowledge, this is the first work which applies federated learning to privacy-preserving native ad CTR prediction with multi-platform user behaviors, and *FedCTR* is the first approach to protect the privacy information in user embeddings at both training and inference stages.

3 METHODOLOGY

In this section, we first introduce the details of our federated native ad CTR prediction method (FedCTR). Then, we introduce the framework to train the FedCTR model where behavior data and model parameters are distributed on different platforms.

3.1 FedCTR for Native Ad CTR Prediction

3.1.1 Overall Framework. The architecture of the proposed FedCTR method is shown in Figure 2. In FedCTR, user behaviors on multiple online platforms are used to infer user interest for native ad CTR prediction, and the behavior data cannot leave its local platforms due to privacy concerns. To solve this problem, in FedCTR we introduce a user server to coordinate different platforms, and the core idea is communicating the intermediate model results and their gradients among server and platforms rather than exchanging the raw user behavior data. We assume that the different platforms have aligned their user IDs via privacy-preserving entity resolution techniques [11, 32]. Concretely, there are three major modules in the FedCTR framework. The first module is ad platform, which is used to predict the CTR scores of a set of native ads using a CTR predictor. It computes the probability score of a target user u clicking a candidate ad d based on their representations $\mathbf{u}$ and $\mathbf{d}$, which is formulated as $\hat{y} = f_{CTR}(\mathbf{u}, \mathbf{d}; \Theta_C)$, where Θ_C represents the parameters of the CTR predictor. The representation of the ad d is computed by an *ad model* based on its ID and text, which is formulated as $\mathbf{d} = f_{ad}(ID, text; \Theta_D)$, where Θ_D denotes the model parameters of the *ad model*. The second module consists of K behavior platforms. Each platform has a *user model* to learn local user representations based on its stored local user behaviors, such as search queries on the search engine platform and browsed webpages on the web browsing platform. For the i-th behavior platform, the learning of local user representation is formulated as $\mathbf{u}_i = f^i_{user}(behaviors; \Theta_{U_i})$, where

Table 1. A summary of different participants for federated native ad CTR prediction.

Participants	Type	Data	Communication
Behavior platforms	Different online platforms (e.g., search engine and ad display website)	User behaviors	Local user embedding (to user server)
Ad platform	Ad display website	Candidate ads, click labels	Prediction query (to user server)
User server	Third-party server to coordinate ad and behavior platforms (not deployed on them)	-	Global user embedding (to ad platform)

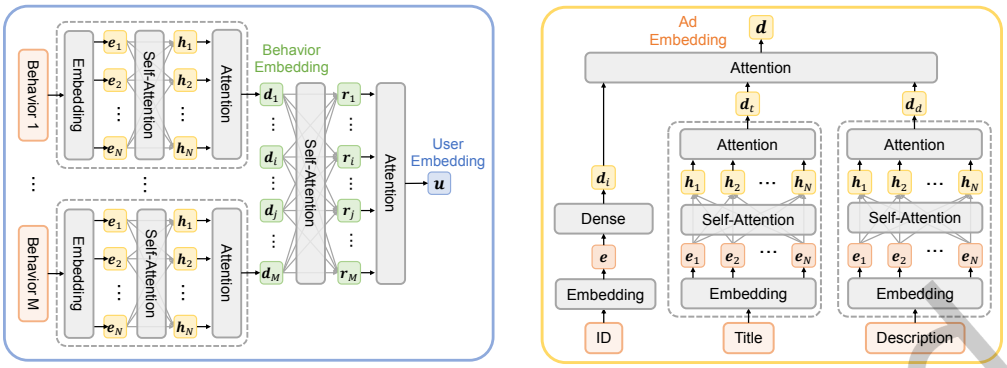
Θ_{U_i} denotes the parameter of the *user model* maintained by this platform. The third module is a user server, which is responsible for coordinating multiple behavior platforms to learn local user embeddings according to the query of the ad platform and aggregating them into a unified user representation $\mathbf{u}$, which is formulated as $\mathbf{u} = f_{agg}(\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K; \Theta_A)$, where Θ_A is the aggregator model parameters. The aggregated user embeddings are further sent to the ad platform. Note that the user server is a third-party one that does not depend on behavior and ad platforms. We summary the characteristics of different participants in *FedCTR* in Table 1. Next, we introduce each module in detail.

In the ad platform, assume there is a set of candidate ads, denoted as $\mathcal{D} = [d_1, d_2, ..., d_P]$. Each ad has an ID, a title and a description. There is an *ad model* in the ad platform that learns representations of ads from their ID, title and description. When a user u visits the website where native ads are displayed, the ad platform is called to compute the personalized CTR scores of the candidate ads for this user. It sends the ID of this user to the user server to query her embeddings inferred from her behaviors on multiple platforms which encode her personalized interest information. When the ad platform receives the user embedding from the user server, it uses a *CTR predictor* to compute the ranking scores of the candidate ads based on user embeddings $\mathbf{u}$ and embeddings of candidate ads $[\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_P]$ using $f_{CTR}(\cdot)$, which are denoted as $[\hat{y}_1, \hat{y}_2, ..., \hat{y}_P]$.

The user server is responsible for user embedding generation by coordinating multiple user behavior platforms. When it receives a user embedding query from the ad platform, it will use the user ID to query the K behavior platforms to learn local user embeddings from local behaviors. After it receives the local user embeddings $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K]$ from different behavior platforms, it uses an *Aggregator* model with the function $f_{agg}(\cdot)$ to aggregate the K local user embeddings into a unified one $\mathbf{u}$, which takes the relative importance of different kinds of behaviors into consideration. Since the unified user embedding $\mathbf{u}$ may still contain some private information of user behaviors, in order to further enhance user privacy protection, we apply differential privacy (DP) technique [7] to $\mathbf{u}$ by adding Laplacian noise with strength λ_{DP} to $\mathbf{u}$. Then the user server sends the perturbed user embedding $\mathbf{u}$ to the ad platform for personalized CTR prediction.

The behavior platforms are responsible for learning user embeddings from their local behaviors. When a behavior platform receives the user embedding query of user u, it will retrieve the behaviors of this user on this platform (e.g., search queries posted to the search engine platform), which are denoted as $[d_1, d_2, ..., d_M]$, where M is the number of behaviors. Then, it uses a neural *user model* to learn the local user embedding $\mathbf{u}_i$ from these behaviors. The user embedding $\mathbf{u}_i$ can capture the user interest information encoded in user behaviors. Since the local user embedding may also contain some private information of the user behaviors on the i-th behavior platform, we apply local differential privacy (LDP) [34] by adding Laplacian noise with strength λ_{LDP} to each local user embedding so as to better protect user privacy. Then, the behavior platform uploads the perturbed local user embedding $\mathbf{u}_i$ to the user server for aggregation.

Next, we provide some discussions on the privacy protection of the proposed FedCTR method. First, in FedCTR the raw user behavior data never leaves the behavior platforms where it is stored, and only the user embeddings



(a) User model. (b) Ad model.

Fig. 3. The architecture of the user and ad models.

learned from multiple behaviors using neural user models are uploaded to user server. According to the data processing inequality [26], the private information conveyed by these local user embeddings is usually much less than the raw user behaviors. Thus, the user privacy can be effectively protected. Second, the user server aggregates the local user embeddings from different platforms into a unified one and sends it to the ad platform. It is very difficult for the ad platform to infer a specific user behavior on a specific platform from this aggregated user embedding. Third, we apply the local differential privacy technique to the local user embeddings on each behavior platform, and apply the differential privacy technique to the aggregated user embedding on user server by adding Laplacian noise for perturbation, making it more difficult to infer the raw user behaviors from the local user embeddings and aggregated user embeddings. Thus, the proposed FedCTR method can well protect user privacy when utilizing user behaviors on different platforms to model user interest for CTR prediction.

3.1.2 Model Details. In this section, we introduce the model details in the FedCTR framework, including the *user model*, *ad model*, *aggregator* and *CTR predictor*.

User Model. User model is used to learn local user embeddings from local user behaviors on the behavior platforms. The user models on different behavior platforms share the same architecture but have different model parameters.[4] The architecture of user model is shown in Figure 3(a). It is based on the neural user model proposed in [41], which learns user embeddings from user behaviors in a hierarchical way. It first learns behavior representations from the texts in behaviors, such as the search query in online search behaviors and the webpage title in webpage browsing behaviors.[5] The behavior representation module first converts the text in behaviors into a sequence of word embeddings. In addition, following [6] we add position embedding to each word embedding to model word orders. Then the behavior representation module uses a multi-head self-attention network [39] to learn contextual word representations by capturing the relatedness among the words in the text. Finally, the behavior representation module applies an attentive pooling network [43] to these contextual word representations which can compute the relative importance of these words and obtain a summarized text representation based on the word representations and their attention weights.

[4]In our approach we use the same hierarchical user model architecture on different platforms for simplicity, but note that FedCTR does not require them to be the same and customized user models are supported.

[5]In our experiments we assume that search queries and browsed webpages are in textual form, and clicked Ads are associated with both texts and IDs. Note that FedCTR can be easily generalized to other types of user behaviors by simply replacing the behavior models for text/ID modeling with other architectures.

After learning the representations of behaviors, a user representation learning module is used to learn user embedding from these behavior embeddings. First, we add a position embedding vector to each behavior embedding vector to capture the sequential order of the behaviors. Then we apply a multi-head self-attention network to learn contextual behavior representations by capturing the relatedness between the behaviors. Finally, we use an attentive pooling network [43] to obtain a unified user embedding vector by summarizing these contextual behavior representations with their attention weights. The model parameters of the user model on the i-th behavior platform are denoted as Θ_{U_i}, and the learning of local user embedding on this platform can be formulated as $\mathbf{u}_i = f^i_{user}(behaviors; \Theta_{U_i})$.

Ad Model. The ad model is used to learn embeddings of ads from their IDs, titles, and descriptions. The architecture of the *ad model* is illustrated in Figure 3(b). It is based on the ad encoder model proposed in [1] with small variants. Similar with the *user model*, we use a combination of word embedding layer, multi-head self-attention layer and attentive pooling layer to learn the embeddings of titles and descriptions from the texts. In addition, we use an ID embedding layer and a dense layer to learn ad representation from ad ID. The final ad representation is learned from the ID embedding, title embedding and description embedding via an attention network [1]. The model parameters of the ad model are denoted as Θ_D, and the learning of ad embedding can be formulated as $\mathbf{d} = f_{ad}(ID, text; \Theta_D)$.

Aggregator. The aggregator model aims to aggregate the local user embeddings learned from different behavior platforms into a unified user embedding for CTR prediction. Since user behaviors on different platforms may have different informativeness for modeling user interest, we use an attention network [43] to evaluate the importance of different local user embeddings when synthesizing them together. It takes the local user embeddings $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K]$ from the K platforms as the input, and learns the aggregated user embedding $\mathbf{u}$ from them via an attention network, which is formulated as follows:

$$\mathbf{u} = f_{agg}(\mathbf{U}; \Theta_A) = \mathbf{U}[\text{softmax}(\mathbf{U}^T \Theta_A)], \tag{1}$$

where Θ_A is the parameters of the aggregator.

CTR Predictor. The CTR predictor aims to estimate the probability score of a user u clicking a candidate ad d based on their representations $\mathbf{u}$ and $\mathbf{d}$, which is formulated as $\hat{y} = f_{CTR}(\mathbf{u}, \mathbf{d}; \Theta_C)$, where Θ_C is the model parameters of the CTR predictor. There are many options for the CTR prediction function $f_{CTR}(\cdot)$, such as dot product [1], outer product [12] and factorization machine [10].

3.2 Federated Model Training

Existing CTR prediction methods usually train the models in a centralized way, where both the training data and the model parameters are located in the same place. In the proposed FedCTR method for native ad CTR prediction, the training data is distributed on multiple platforms. For example, the ad information and the users' click and non-click behaviors on ads which can serve as labels for model training are located in the ad platform, while users' behaviors on many online platforms are located in other platforms, which cannot be centralized due to privacy constraints. In addition, FedCTR contains multiple models such as user models, ad model, and aggregator model which are distributed on different platforms. Thus, it is a non-trivial task to train the models of FedCTR without violating the privacy protection requirement. Motivate by [26], in this section we present a privacy-preserving framework to train the models for FedCTR where each platform learns the model on it in a federated way, and only the gradients of user embeddings (rather than raw user behaviors or models) are communicated across different platforms. The framework for model training is shown in Figure 4, and the data communications among different participants are summarized in Table 2. The details of model training are introduced as follows.

At the model training stage, for a randomly selected user behavior on the ad platform denoted as (u, d, y, t) which means at timestamp t an ad d is displayed to user u and her click behavior is y (1 for click and 0 for non-click), we first use the FedCTR framework to learn the local user embeddings $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K]$ and the aggregated

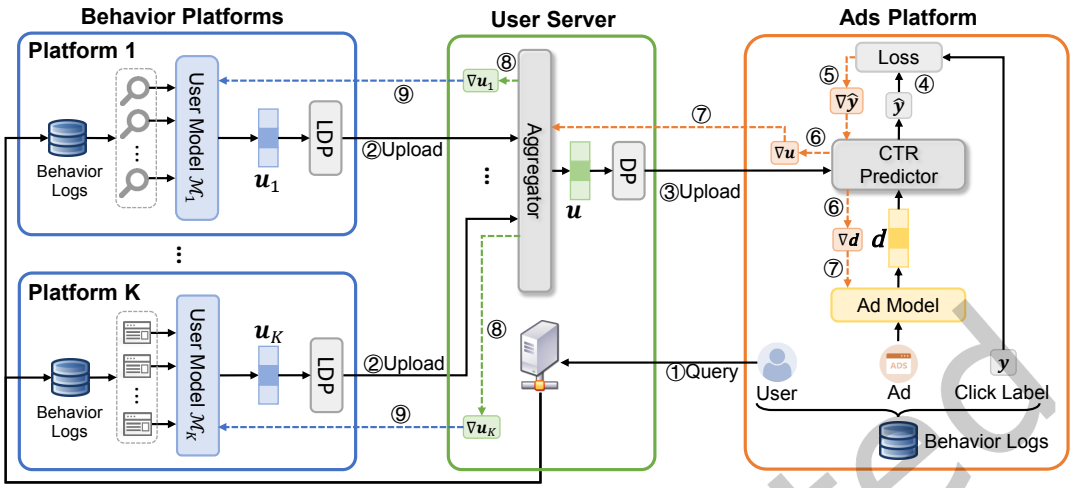


Fig. 4. The model training framework for our *FedCTR* approach.

Table 2. A summary of different participants for federated model training.

Participants	Communication
Behavior platforms	local user embedding (to user server)
Ad platform	prediction query (to user server), gradients of global user embedding (to user server)
User server	global user embedding (to ad platform), gradients of local user embeddings (to behavior platforms)

user embedding $\mathbf{u}$ at timestamp t using the current user and aggregator models (steps 1-3). We also use the current ad model to learn an embedding $\mathbf{d}$ of this ad. Then we use the current CTR predictor model to compute the predicted click probability score $\hat{y}$. By comparing $\hat{y}$ with y, we can compute the loss of the current FedCTR models on this training sample (step 4). In our model training framework cross-entropy loss is used, and loss of this sample can be formulated as:

$$\mathcal{L}(u, d; \Theta_D, \Theta_C, \Theta_A, \Theta_U) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \tag{2}$$

where $\Theta_U = [\Theta_{U_1}, \Theta_{U_2}, ..., \Theta_{U_K}]$ is the parameter set of user models on different platforms.

Then we compute the model gradients for model update (steps 5-9). According to the loss $\mathcal{L}$, we compute the gradient of $\hat{y}$, denoted as $\nabla\hat{y}$. We input $\nabla\hat{y}$ to the CTR predictor. Based on $\nabla\hat{y}$ and the CTR prediction function $\hat{y} = f_{CTR}(\mathbf{u}, \mathbf{d}; \Theta_C)$, we can compute the gradient of parameters Θ_C in CTR predictor (denoted as $\nabla\Theta_C$), the gradient of user embedding $\mathbf{u}$ (denoted as $\nabla\mathbf{u}$) and the gradient of ad embedding $\mathbf{d}$ (denoted as $\nabla\mathbf{d}$). The model parameters of the CTR predictor Θ_C can be updated using $\nabla\Theta_C$ following the SGD [2] algorithm, i.e., $\Theta_C = \Theta_C - \eta\nabla\Theta_C$, where η is the learning rate.

The gradient of ad embedding $\nabla\mathbf{d}$ is then sent to the ad model. Based on $\nabla\mathbf{d}$ and the function $\mathbf{d} = f_{ad}(ID, text; \Theta_D)$ that summarizes the neural architecture of *ad model*, we can compute the gradient of the ad model, denoted as $\nabla\Theta_D$. We use $\nabla\Theta_D$ to update the model parameters of ad model Θ_D, i.e., $\Theta_D = \Theta_D - \eta\nabla\Theta_D$.

Algorithm 1 FedCTR

Initialize Θ_D and Θ_C on the Ad platform, Θ_A on the user server and Θ_{U_i} on behavior platforms

// Model Training

repeat
  Ad platform randomly selects a subset $\mathcal{S}$ from the user set $\mathcal{U}$
  Ad platform generates candidate ads set $\mathcal{D}$.
  $\hat{y} \leftarrow$ **FedCTR_Prediction**($\mathcal{S}$, $\mathcal{D}$)
  Ad platform computes loss $\mathcal{L}$ based on $\hat{y}$ and click labels
  Ad platform computes $\nabla\hat{y}$, $\nabla\mathbf{u}$, and $\mathbf{d}$ based on f_{CTR}
  Ad platform updates CTR predictor and Ad model
  Ad platform sends $\nabla\mathbf{u}$ to the user server
  User server computes $[\nabla\mathbf{u}_1, \nabla\mathbf{u}_2, ..., \nabla\mathbf{u}_K]$ based on f_{agg}.
  User server sends $\nabla\mathbf{u}_i$ to the i-th behavior platform
  Each behavior platform updates local user model based on f^i_{user}
until model convergence

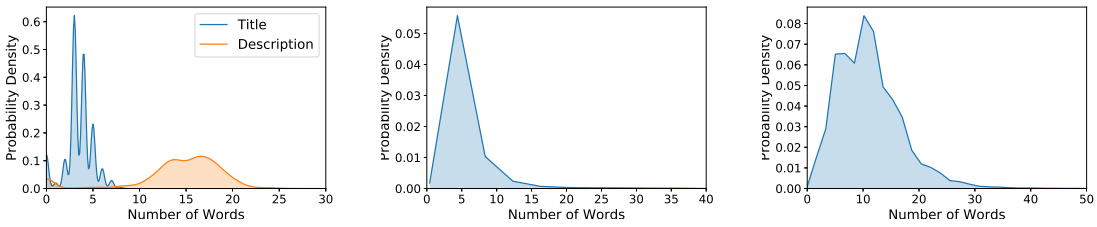
// Prediction

FedCTR_Prediction($\mathcal{S}$, $\mathcal{D}$):
Ad platform queries the user server to compute the embeddings of users in $\mathcal{S}$
User server calls each behavior platform to compute a local embedding $\mathbf{u}_i$ of each user in $\mathcal{S}$ based on local behaviors
Behavior platforms upload the local user embeddings to the user server
User server aggregates the local user embeddings $[\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_K]$ of each user in $\mathcal{S}$ into $\mathbf{u}$.
User server sends $\mathbf{u}$ to the Ad platform
Ad platform uses $\hat{y} = f_{CTR}$ to conduct CTR prediction
return $\hat{y}$

The user embedding gradient $\nabla\mathbf{u}$ is distributed to the aggregator in the user server. Based on the aggregator model function $\mathbf{u} = f_{agg}(\mathbf{U}; \Theta_A)$ and $\nabla\mathbf{u}$, we compute the gradients of the aggregator model (denoted as $\nabla\Theta_A$) as well as the gradient of each local user embedding (denoted as $\nabla\mathbf{u}_i, i = 1, ..., K$). The model parameters of the aggregator can be updated as $\Theta_A = \Theta_A - \eta\nabla\Theta_A$. The gradients of the local embeddings, i.e., $\nabla\mathbf{u}_i, i = 1, ..., K$, are sent to the corresponding behavior platform respectively.

On each behavior platform, when it receives the gradient of the local user embedding, the platform will combine the local user embedding gradient, the input user behaviors and the current user model based on the function $\mathbf{u}_i = f^i_{user}(behaviors; \Theta_{U_i})$ to compute the gradient of the user model, which is denoted as $\nabla\Theta_{U_i}$ on the i-th platform. Then the model parameters of the local user models are updated as $\Theta_{U_i} = \Theta_{U_i} - \eta\nabla\Theta_{U_i}$. Above model training process is conducted on different training samples for multiple rounds until models converge. We summarize the process of our *FedCTR* method in Algorithm 1.

In our federated model training framework, the user behaviors on different platforms (e.g., the ad platform and the multiple behavior platforms for user interest modeling) never leave the local platforms, and only the model gradients are distributed from the ad platform to user server, and from user server to each user behavior platform. Since model gradients usually contain much less private information than the raw user behaviors [26], user privacy can be protected at the training stage of the proposed FedCTR method.



(a) Ad title and description. (b) Search query. (c) Webpage title.

Fig. 5. Distributions of ad title, ad description, search query and webpage title lengths.

4 EXPERIMENTS

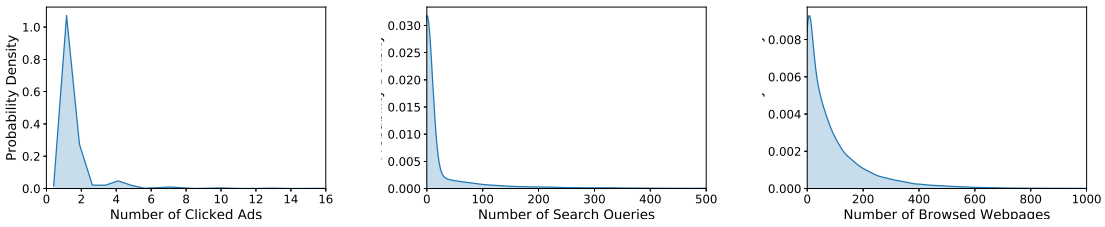
4.1 Datasets and Experimental Settings

Since there is no publicly available dataset for native ad CTR prediction, we constructed one by collecting the logs of 100,000 users on the native ads displayed on the Bing Ads platform from 11/06/2019 to 02/06/2020. The logs in the last week were reserved for testing, and the remaining logs were used for model training. We randomly selected 10% of the samples in training set for validation. We also collected the search logs and webpage browsing behavior logs of users recorded by a commercial search engine during the same period for user interest modeling. The detailed statistics of the dataset are shown in Table 3. We assume that different kinds of user behavior data come from independent platforms and they cannot be directly aggregated. In this way, the behavior platforms include (1) the search engine that keeps users' search behaviors; (2) the web browser platform that keeps users' webpage behaviors; (3) the ad display platform that records users' ad click behaviors (the ad platform itself also serves as a behavior platform).

Table 3. Detailed statistics of the dataset for native ad CTR prediction. * We sample the same number of non-click behaviors with click behaviors for balanced model training.

#users	100,000	avg. #words per ad title	3.73
#ads	8,105	avg. #words per ad description	15.31
#ad click behaviors	345,264	avg. #words per search query	4.64
#ad non-click behaviors*	345,264	avg. #words per webpage title	10.84
avg. #queries per user	50.69	avg. #webpages per user	210.09

In our experiments, the word embeddings in the user and ad models were initialized by the pre-trained Glove [30] embeddings. The ad ID embeddings are randomly initialized. In the CTR predictor, following [1] we used dot-product as the CTR prediction function. Each self-attention networks had 16 heads, and the output from each head was 16-dimensional, which means that the total hidden dimension is 256. The query vectors in attention networks were also 256-dimensional. Adam [16] was used as the model optimizer (with a learning rate of 1e-3). The batch size was set to 32. The number of epoch is 2. The dropout [38] ratio after each layer was 20%. The strength of Laplace noise λ_{LDP} in the LDP modules was 0.01, and λ_{DP} in the DP module was 0.005. Hyperparamters were tuned on the validation set. To evaluate the model performance, on the *Ads* dataset we used AUC and AP as the metrics. We reported the average results of 10 independent experiments.



(a) Clicked Ad. (b) Search query. (c) Browsed webpage.

Fig. 6. Distributions of the numbers of historical ad clicking, search query and webpage browsing behaviors of users.

4.2 CTR Prediction Performance

We compare the performance of *FedCTR* with several baseline methods, include:

- LR [3, 37], logistic regression, a widely used method for ads CTR prediction. We used ad IDs and the TF-IDF features extracted from the texts of behaviors and ads as the input.
- FM [35], factorization machine, which is a popular method for CTR prediction. We used the same features as LR.
- Wide&Deep [5], a popular CTR prediction method with a wide linear part and a deep neural part. Same features with LR were used.
- PNN [33], product-based neural network for CTR prediction which can model the interactions between features.
- DSSM [13], deep structured semantic model, a famous model for CTR prediction and recommendation.
- DeepFM [10], a combination of factorization machines and deep neural networks for CTR prediction.
- DIN [45], deep interest network for CTR prediction. We use the same ad and behavior models with our approach to learn ad and behavior features.
- DIEN [44], deep interest evolution network for CTR prediction. Same ad and behavior features with DIN are used.
- DIFM [23], a dual input-aware factorization machine for CTR Prediction. Same ad and behavior features are used.
- NativeCTR [1], a neural native ads CTR prediction method based on attentive multi-view learning.

We report the performance of baseline methods based on user behaviors (i.e., ad clicks) on the ads platform only and their ideal performance using the centralized storage of behavior data from different platforms. The results under different ratios of training data of different methods are shown in Table 4. According to the results, we find the methods using neural networks to learn user and ad representations (e.g., *FedCTR*) perform better than those using handcrafted features to represent users and ads (e.g., *LR* and *FM*). It implies that the representations learned by neural networks are more suitable than handcrafted features in modeling users and ads. In addition, compared with the methods solely based on ad click behaviors on the ad platform for user interest modeling, the methods that consider multi-platform behaviors in user modeling can achieve better performance. This is because the user behavior data on the ad platform may be sparse, which is insufficient to infer user interest accurately. Since user behaviors on different platforms can provide rich clues for inferring user interest in different aspects. Unfortunately, user behavior data is highly privacy-sensitive and usually cannot be centrally stored or exchanged among platforms due to the constraints of data protection regulations like GDPR and the privacy concerns from users. Thus, in many situations the methods that need the centralized storage of multi-platform user behavior data

Table 4. Comparisons of different methods. * means the ideal performance under centralized user behavior data from different platforms and centralized model learning.

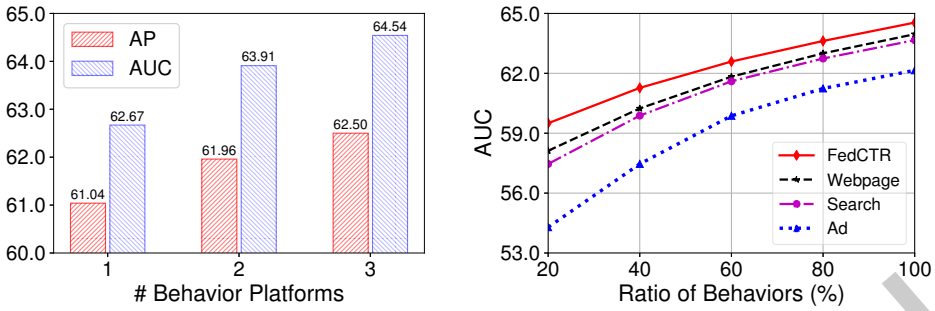
Methods	25%		50%		100%	
	AUC	AP	AUC	AP	AUC	AP
LR[37]	58.42	56.47	58.96	56.87	59.36	57.24
LR*	60.82	57.38	61.44	58.60	62.04	59.20
FM[35]	57.79	55.91	58.10	56.33	58.45	56.71
FM*	61.59	58.13	61.91	59.17	62.47	59.99
Wide&Deep[5]	59.45	57.80	59.66	57.97	60.04	58.61
Wide&Deep*	62.10	59.75	62.35	59.87	62.79	60.28
PNN[33]	59.53	57.86	59.73	58.03	60.02	58.50
PNN*	62.54	60.12	62.73	60.29	62.87	60.41
DSSM[13]	59.24	57.59	59.46	57.89	59.92	58.43
DSSM*	62.23	59.66	62.50	59.85	62.85	60.37
DeepFM[10]	59.36	57.70	59.55	57.92	59.83	58.28
DeepFM*	61.88	59.47	62.05	59.77	62.72	60.24
DIN[45]	60.91	59.22	61.32	59.43	61.52	60.04
DIN*	63.05	60.77	63.03	60.89	63.58	61.35
DIEN[44]	60.96	59.42	61.39	59.59	61.71	60.13
DIEN*	63.17	60.89	63.20	60.95	63.61	61.44
DIFM [23]	60.90	59.34	61.34	59.55	61.66	60.08
DIFM*	63.02	60.78	63.16	60.90	63.55	61.39
NativeCTR[1]	60.84	59.12	61.12	59.40	61.44	59.75
NativeCTR*	62.88	60.69	63.01	60.88	63.39	61.17
FedCTR	**63.95**	**61.82**	**64.20**	**62.13**	**64.54**	**62.50**

may not achieve the ideal performance in Table 4. Besides, our *FedCTR* method consistently outperforms other baseline methods. This is because our framework is more effective in leveraging multi-platform user behaviors for user interest modeling than other baseline methods, and the multi-head self-attention networks in the ad and user models also have greater ability in learning ad and user representations than other architectures like FM, CNN and RNN. Moreover, in our approach the raw behavior data never leaves the local platforms, and only user embeddings and model gradients are communicated among different platforms. Thus, our approach can model user interest in a privacy-preserving manner.

4.3 Effect of Multi-Platform Behaviors

Next, we explore the effectiveness of incorporating user behavior data from multiple platforms for CTR prediction. We first study the influence of the number of behavior platforms for user modeling. The average results of *FedCTR* with different numbers of platforms are shown in Figure 7(a). From the results, we find that the performance improves as number of platforms increases. This is probably because user behaviors on different platforms can provide complementary information to help cover user interest more comprehensively. It shows that incorporating multi-platform user behaviors can effectively enhance user interest modeling for CTR prediction.

We also study the influence of the number of user behaviors on each platform on the CTR prediction performance. We vary the ratios of user behaviors for user interest modeling from 20% to 100%, and the results are shown in Figure 7(b). From the results, we find that the performance of *FedCTR* declines when the number of



(a) Influence of the number of behavior platform for user modeling.

(b) Influence of the number of user behaviors.

Fig. 7. Effect of multi-platform user behaviors.

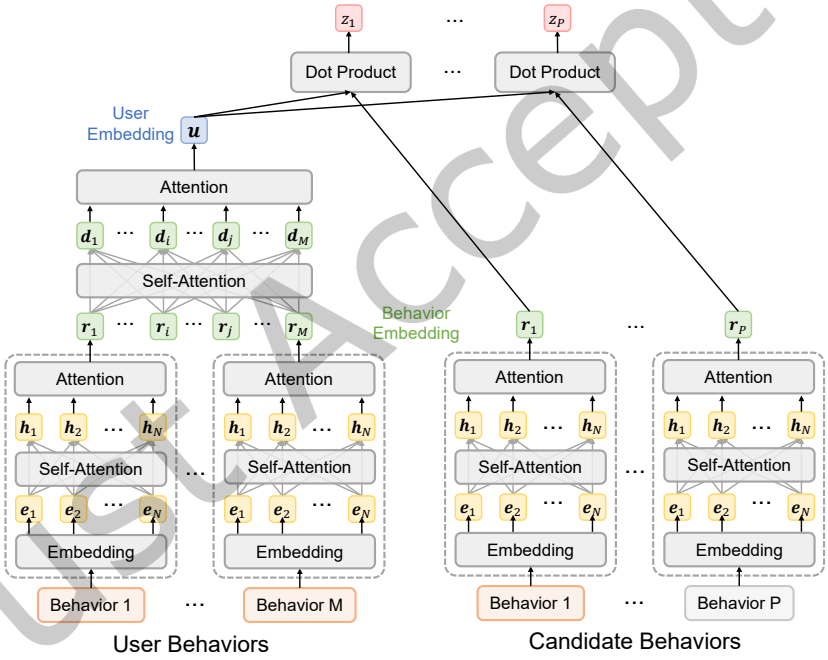
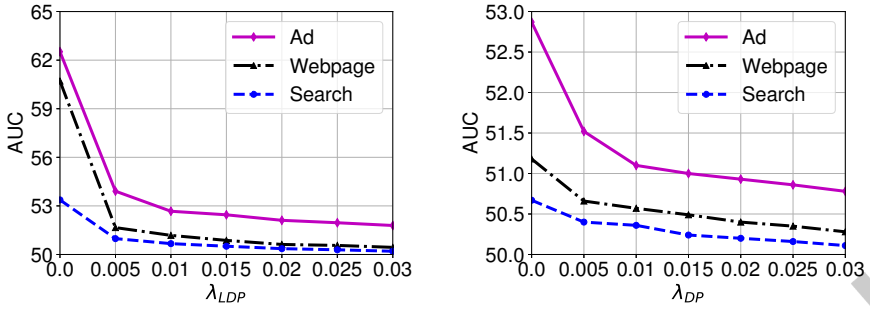


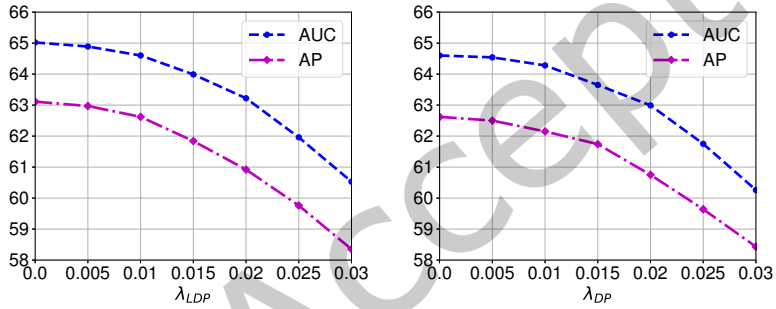
Fig. 8. The evaluation model framework in our privacy protection experiments.

behaviors decreases. It indicates that it is more difficult to infer user interest when user behaviors are scarce. In addition, we find that the *FedCTR* method consistently outperforms its variants with single-platform user behaviors, and the advantage becomes larger when user behaviors are scarcer. It shows that utilizing the user behavior data decentralized on different platforms can help model user interest more accurately, especially when user behaviors on a single platform are sparse.



(a) Local embedding. (b) Aggregated embedding.

Fig. 9. Privacy protection of local and aggregated user embedding under different λ_{LDP} and λ_{DP} values. Lower AUC indicates better privacy protection.



(a) λ_{LDP}. (b) λ_{DP} .

Fig. 10. Influence of λ_{LDP} and λ_{DP} on CTR prediction.

4.4 Study on Privacy Protection

In this section, we verify the effectiveness of our *FedCTR* method in privacy protection when exploiting multi-platform user behavior data for user modeling. Since user embeddings learned from different platforms may contain private information that can be used to infer the raw behavior data [24], we use LDP and DP to protect the local and aggregated user embeddings, respectively. In our experiments, we observe that the sensitivity of local user embeddings is about 0.05, which means that their privacy budget ϵ is 5. The privacy budget of the aggregated user embedding is smaller because more noise is added. To empirically evaluate the privacy protection performance of these embeddings, we use a behavior prediction task by predicting the raw user behaviors from the local and aggregated user embeddings to indicate their privacy protection ability. The evaluation framework is shown in Fig. 8. More specifically, for each user we randomly sample a real behavior of this user (regarded as a positive sample) and $P-1$ behaviors which do not belong to this user (regarded as negative samples).[6] The goal is to infer the real behavior from these P candidate behaviors by measuring their similarities to the user embedding. We perform dot product between the embeddings of each user-behavior pair to compute relevance

[6]We randomly select 9 negative samples for each positive sample.

scores $[z_1, z_2, ..., z_P]$, then all candidate behaviors are ranked according to the relevance scores.[7] We use AUC as the metric to evaluate the privacy protection performance, and lower scores mean better privacy protection.

There are two key hyperparameters in the LDP and DP modules, i.e., λ_{LDP} and λ_{DP}, which control the strength of the Laplacian noise added to user embeddings. We first vary the value of λ_{LDP} to explore its influence on privacy protection and CTR prediction, and the results are shown in Figure 9(a) and 10(a), respectively.[8] We find that although the CTR prediction performance is slightly better if the local user embeddings are not protected by LDP, the raw user behaviors can be inferred to a certain extent, which indicates that the private information of local user embeddings is not fully protected. Thus, it is important to use LDP to protect the local embeddings learned from different behavior platforms. In addition, if λ_{LDP} is too large, the CTR performance declines significantly, and the improvement on privacy protection is marginal. Thus, we choose a moderate λ_{LDP} (i.e., 0.01) to achieve a trade-off between CTR prediction and privacy protection. Then, we explore the influence of λ_{DP} on the performance of *FedCTR* in privacy protection and CTR prediction (under $\lambda_{LDP} = 0.01$), and the results are respectively shown in Figure 9(b) and 10(b). We find it is also important to set a moderate value for λ_{DP} (e.g., 0.005) to balance the performance of CTR prediction and privacy protection. Besides, comparing the privacy protecting performance on local and aggregated embeddings, we find that it is more difficult to infer the raw user behaviors on a specific platform from the aggregated user embedding than from local user embeddings. We hypothesise this may be because the aggregated user embedding is a summarization of local embeddings, thus the private information is more difficult to be recovered.

4.5 Study on Computational and Communication Cost

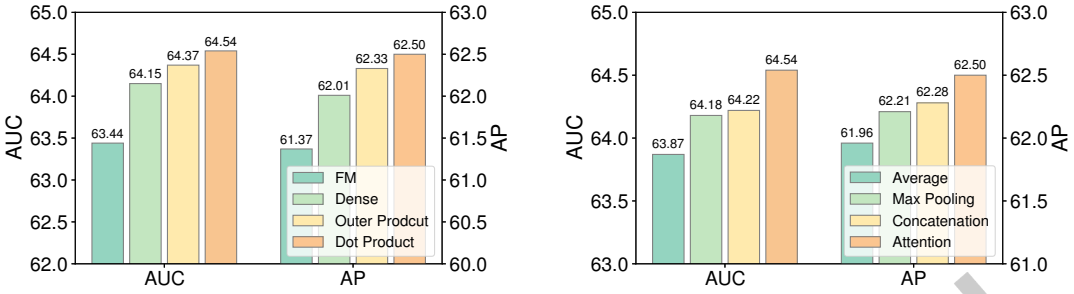
In this section, we present some analysis of the computational and communication cost of *FedCTR*. Since the self-attention network has quadratic complexity with respect to the input sequence length, the total theoretical computational cost of CTR prediction on each platform is $O(M^2 + MN^2)$, where N is the number of words in a user behavior and M is the number of the behaviors of a user on this platform. In practice, the local training time takes around 2300 seconds per epoch (2 epochs in total). In our *FedCTR* model, there are 88.56M parameters in total. Among them, the word embeddings on different platforms take 84.92M parameters, and the Ad ID embeddings have 2.07 parameters. In the model training process, there are 21,580 rounds of communications among the user server and behavior platforms. The overall communication cost in the model training process is 168.59MB (all parameters are in 32-bit float format), which is mainly handled by the user server. The size of communicated data is quite acceptable for the behavior platforms and user server which usually have adequate communication bandwidths.

4.6 Effect of Aggregator and CTR Predictor

We also verify the effectiveness of the aggregator and CTR predictor models. First, we compare different CTR prediction models like factorization machine (FM) [10], dense layers, outer product [12] and dot product [1], and the results are shown in Figure 11(a). From Figure 11(a), we find the performance of FM is not optimal. It shows that FM may not be suitable for modeling the similarity between the user and ad representations learned by neural networks. In addition, we find that using a dense layer is also not optimal. This may be because dense layers compute the click scores based on the concatenation of ad and user representations while difficult to model their interactions, which is also validated by [36]. Besides, it is interesting that dot product achieves the best performance. This may be because dot product simultaneously models the distance between two vectors as well as their lengths, which can effectively measure the relevance of user and ad representations for CTR prediction. Thus, we prefer dot product for its effectiveness and simplicity.

[7]Note that the model parameters are frozen.

[8]The DP module is deactivated in these experiments.



(a) CTR predictor.

(b) Aggregator.

Fig. 11. Different CTR predictor and aggregator models.

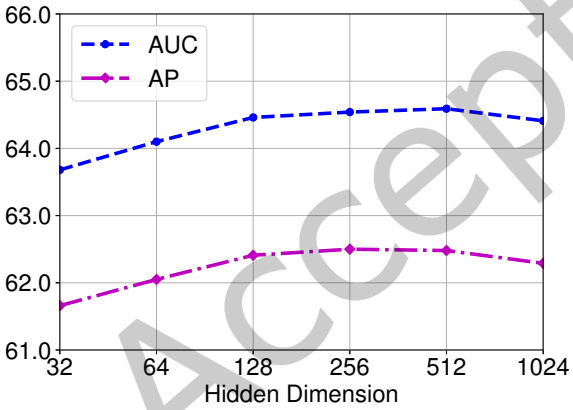


Fig. 12. Influence of hidden dimensions of ad and user models on CTR prediction performance.

Then, we compare different models for user embedding aggregation, including attention network, average pooling, max pooling and concatenation. The results are shown in Figure 11(b). We find that average pooling is sub-optimal for aggregation, since it cannot distinguish the informativeness of different local user embeddings. In addition, max pooling is also not optimal, since it only keeps the most salient features. Moreover, although concatenating user embeddings can keep more information, it is inferior to using attention mechanism due to its lack of informativeness modeling. Thus, we use attention networks to implement the aggregator in the user server.

4.7 Influence of Model Size

Finally, we study the influence of model size on the CTR prediction performance. We change the model size by varying the hidden dimension in the ad and user models, and the corresponding model performance is shown in Fig. 12. We find that the model performance first improves with the increase of hidden dimension of ad and user models, while the performance does not significantly improve or even declines if the hidden dimension is larger than 256. Thus, in *FedCTR* we set the hidden dimension to 256 to achieve the best performance.

5 CONCLUSION AND FUTURE WORKS

In this paper, we propose a federated native ad CTR prediction method based on multi-platform user behaviors. In our method, each platform learns local user embeddings from the local user behavior data, and upload them to a user server for aggregation. The aggregated user embedding is sent to the ad platform for CTR prediction. In addition, we apply LDP and DP techniques to the local and aggregated user embeddings respectively to better protect user privacy. Besides, we propose a federated model training framework to coordinate different platforms to collaboratively train the models of FedCTR by sharing model gradients rather than raw behaviors to protect user privacy at the model training stage. Experiments on a real-world dataset show that FedCTR is effective in native ad CTR prediction by incorporating multi-platform user behaviors with user privacy well-protected.

In future, we plan to deploy FedCTR to online ad system and test its online performance. We are also interested in applying FedCTR to enhance other CTR prediction tasks (e.g., search ads) by improving their user modeling part via incorporating multi-platform user behavior data in a privacy-preserving way.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Program of China under Grant number 2018YFC1604000/2018YFC1604002.

REFERENCES

[1] Mingxiao An, Fangzhao Wu, Heyuan Wang, Tao Di, Jianqiang Huang, and Xing Xie. 2019. Neural CTR Prediction for Native Ad. In *CCL*. Springer, 600–612.
[2] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*. Springer, 177–186.
[3] Deepayan Chakrabarti, Deepak Agarwal, and Vanja Josifovski. 2008. Contextual advertising by combining relevance with click feedback. In *WWW*. 417–426.
[4] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua. 2016. Deep ctr prediction in display advertising. In *MM*. 811–820.
[5] 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 *DLRS*. ACM, 7–10.
[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.
[7] Cynthia Dwork. 2008. Differential privacy: A survey of results. In *TAMC*. Springer, 1–19.
[8] Siwei Feng and Han Yu. 2020. Multi-Participant Multi-Class Vertical Federated Learning. *arXiv preprint arXiv:2001.11154* (2020).
[9] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. In *AAAI*. 2301–2307.
[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *AAAI*. 1725–1731.
[11] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).
[12] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer product-based neural collaborative filtering. In *IJCAI*. 2227–2233.
[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. ACM, 2333–2338.
[14] Di Jiang, Yuanfeng Song, Yongxin Tong, Xueyang Wu, Weiwei Zhao, Qian Xu, and Qiang Yang. 2019. Federated Topic Modeling. In *CIKM*. 1071–1080.
[15] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *RecSys*. 43–50.
[16] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.
[17] Feng Li, Zhenrui Chen, Pengjie Wang, Yi Ren, Di Zhang, and Xiaoyu Zhu. 2019. Graph Intention Network for Click-through Rate Prediction in Sponsored Search. In *SIGIR*. 961–964.
[18] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable Click-Through Rate Prediction through Hierarchical Attention. In *WSDM*. 313–321.

[19] 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 *CIKM*. 539–548.
[20] 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 *KDD*. 1754–1763.
[21] Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. 2020. AutoGroup: Automatic feature grouping for modelling explicit high-order feature interactions in CTR prediction. In *SIGIR*. 199–208.
[22] Yang Liu, Yan Kang, Xinwei Zhang, Liping Li, Yong Cheng, Tianjian Chen, Mingyi Hong, and Qiang Yang. 2019. A Communication Efficient Vertical Federated Learning Framework. *arXiv preprint arXiv:1912.11187* (2019).
[23] Wantong Lu, Yantao Yu, Yongzhe Chang, Zhen Wang, Chenhui Li, and Bo Yuan. 2020. A Dual Input-aware Factorization Machine for CTR Prediction.. In *IJCAI*. 3139–3145.
[24] Lingjuan Lyu, Yitong Li, Xuanli He, and Tong Xiao. 2020. Towards Differentially Private Text Representations. In *SIGIR*. 1813–1816.
[25] Stéphane Matteo and Cinzia Dal Zotto. 2015. Native advertising, or how to stretch editorial to sponsored content within a transmedia branding era. In *Handbook of media branding*. Springer, 169–185.
[26] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*. 1273–1282.
[27] Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2018. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035* (2018).
[28] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *WWW*. 1349–1357.
[29] Mehul Parsana, Krishna Poola, Yajun Wang, and Zhiguang Wang. 2018. Improving native ads ctr prediction by large scale event embedding and recurrent networks. *arXiv preprint arXiv:1804.09133* (2018).
[30] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.
[31] 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 *KDD*. 2671–2679.
[32] Benny Pinkas, Thomas Schneider, and Michael Zohner. 2014. Faster private set intersection based on {OT} extension. In {*USENIX*} *Security*. 797–812.
[33] 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 *ICDM*. IEEE, 1149–1154.
[34] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A McCann, and S Yu Philip. 2018. High-Dimensional Crowdsourced Data Publication with Local Differential Privacy. *TIFS* (2018), 2151–2166.
[35] Steffen Rendle. 2012. Factorization machines with libfm. *TIST* 3, 3 (2012), 57.
[36] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *RecSys*. 240–248.
[37] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. 521–530.
[38] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[40] Bartosz W Wojdynski and Nathaniel J Evans. 2016. Going native: Effects of disclosure position and language on the recognition and evaluation of online native advertising. *Journal of Advertising* 45, 2 (2016), 157–168.
[41] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural News Recommendation with Multi-Head Self-Attention. In *EMNLP*. 6390–6395.
[42] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *TIST* 10, 2 (2019), 1–19.
[43] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*. 1480–1489.
[44] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *AAAI*, Vol. 33. 5941–5948.
[45] 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 *KDD*. 1059–1068.