

# Image Matters: Visually Modeling User Behaviors Using Advanced Model Server

Tiezheng Ge, Liqin Zhao, Guorui Zhou, Keyu Chen, Shuying Liu  
Huimin Yi, Zelin Hu, Bochao Liu, Peng Sun, Haoyu Liu, Pengtao Yi, Sui Huang  
Zhiqiang Zhang, Xiaoqiang Zhu, Yu Zhang, Kun Gai  
Alibaba Inc.  
{tiezheng.gtz, zhang.zhiqiang, jingshi.gk}@alibaba-inc.com

## ABSTRACT

In Taobao, the largest e-commerce platform in China, billions of items are provided and typically displayed with their images. For better user experience and business effectiveness, Click Through Rate (CTR) prediction in online advertising system exploits abundant user historical behaviors to identify whether a user is interested in a candidate ad. Enhancing behavior representations with user behavior images will help understand user's visual preference and improve the accuracy of CTR prediction greatly. So we propose to model user preference jointly with user behavior ID features and behavior images. However, training with user behavior images brings tens to hundreds of images in one sample, giving rise to a great challenge in both communication and computation. To handle these challenges, we propose a novel and efficient distributed machine learning paradigm called Advanced Model Server (AMS). With the well-known Parameter Server (PS) framework, each server node handles a separate part of parameters and updates them independently. AMS goes beyond this and is designed to be capable of learning a unified image descriptor model shared by all server nodes which embeds large images into low dimensional high level features before transmitting images to worker nodes. AMS thus dramatically reduces the communication load and enables the arduous joint training process. Based on AMS, the methods of effectively combining the images and ID features are carefully studied, and then we propose a Deep Image CTR Model. Our approach is shown to achieve significant improvements in both online and offline evaluations, and has been deployed in Taobao display advertising system serving the main traffic.

## CCS CONCEPTS

• **Information systems** → **Online advertising; Recommender systems;**

## KEYWORDS

Online advertising; User modeling; Computer vision

## ACM Reference Format:

Tiezheng Ge, Liqin Zhao, Guorui Zhou, Keyu Chen, Shuying Liu, Huimin Yi, Zelin Hu, Bochao Liu, Peng Sun, Haoyu Liu, Pengtao Yi, Sui Huang, Zhiqiang Zhang, Xiaoqiang Zhu, Yu Zhang, Kun Gai, Alibaba Inc., {tiezheng.gtz, zhang.zhiqiang, jingshi.gk}@alibaba-inc.com. 2018. Image Matters: Visually Modeling User Behaviors Using Advanced Model Server. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*, October 22–26, 2018, Torino, Italy. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3269206.3272007>

## 1 INTRODUCTION

Taobao is the largest e-commerce platform in China, serving hundreds of millions of users with billions of items through both mobile app and PC website. Users come to Taobao to browse these items through the search or personalized recommendation. Each item is usually displayed by an item image along with some describing texts. When interested in an item, users click that image to see the details. Fig 1(a) shows an example of recommended items in Taobao mobile app.

Taobao also established one of the world's leading display advertising systems, helping millions of advertisers connect to customers. By identifying user interests, display ads are presented in various spots like Guess What You Like and efficiently deliver marketing messages to the right customers. Cost-per-click (CPC) pricing method is adopted and sufficiently effective [32]. In CPC mode, the ad publishers rank the candidate ads in order of effective cost per mille (eCPM), which can be estimated by the product of the bid price and the estimated click through rate (CTR). Such strategy makes CTR prediction the core task in the advertising system.

CTR prediction scores a user's preference to an item, and largely relies on understanding user interests from historical behaviors. Users browse and click items billions of times in Taobao every day, and these visits bring a huge amount of log data which weakly reflect user interests. Traditional researches on CTR prediction focus on carefully designed feedback feature [1, 28] and shallow models, e.g., Logistic Regression [23]. In recent years, the deep learning based CTR prediction system emerged overwhelmingly [30]. These methods mainly involve the sparse ID features, e.g., ad ID, user interacted item ID, etc. However, when an ID occurs less frequently in the data, its parameter may not be well trained. Images can provide intrinsic visual descriptions, and thus bring better generalization for the model. Considering that item images are what users directly interact with, these images can provide more visual information about user interests. We propose to naturally describe each behavior by such images, and jointly model them with ID features in CTR prediction.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

CIKM '18, October 22–26, 2018, Torino, Italy

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6014-2/18/10...\$15.00

<https://doi.org/10.1145/3269206.3272007>

Training CTR models with image data requires huge computation and storage consumption. There are pioneering works [3, 21] dedicating to represent ad with image features in CTR prediction. These studies did not explore user behavior images. Modeling user behavior images can help understand user visual preference and improve the accuracy of CTR prediction. Moreover, combining both user visual preference and ad visual information could further benefit CTR prediction. However, modeling user preference with interacted images is more challenging. Because the number of one typical user's behaviors ranges from tens to hundreds, it will bring the same number of times the consumption than that when only modeling ad images, and is a non-trivial problem. A well designed efficient training system is a must to handle this large scale problem for real production.

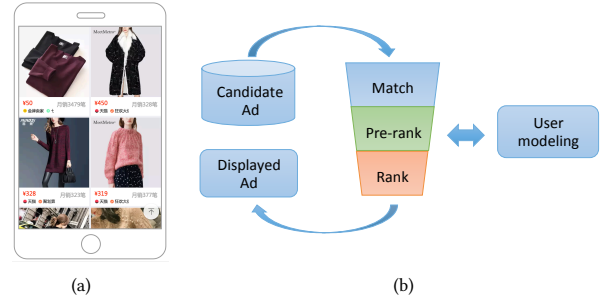
We propose Advanced Model Server (AMS) framework, which goes beyond the well-known Parameter Server (PS) [17, 25] to handle this large scale training problem. The key motivation is to learn a unified high level image descriptor among all server nodes, which embeds large images into low dimensional features before transmitting raw images to workers. In this way, not only can the communication load be largely reduced, but also the repetitive computation of descriptor can be aggregated in servers. With traditional PS framework, however, each of the server nodes handles a separate part of parameters and updates them independently. Thus PS lacks the ability to learn the unified image descriptor which is shared among all server nodes. In AMS, servers are designed to be capable of forwarding and globally updating a shared learnable sub-model with images distributed on each server node.

With this design, the whole CTR model is divided into the worker model, which assembles all features to predict CTR on workers, and the server model, which is the high level image descriptor learned on servers. Then, raw image feature data are distributed on server side as globally shared features without repetition, largely reducing storage usage than the in-sample storage, in our application, by about 31 times. And only low dimensional high-level semantic representations of images output by the server model part rather than raw images need to be transmitted, dramatically reducing communication load, in our application, by about 32 times. Moreover, gradients are back propagated completely from worker model to server model, which guarantees an end-to-end training from raw image features to the final CTR score.

Based on AMS, we successfully build a highly efficient training system and deploy a light weight online service, which manages to tackle the heavy load of storage, computation and communication brought by the image features. Specifically, our training process with billions of samples finishes in 18 hours, enabling the daily update of online model, a required character for industrial production.

Benefitting from the carefully optimized infrastructure, we propose a unified network architecture, named Deep Image CTR Model (DICM) that effectively models users with behavior images. DICM achieves image aware user modeling through a selected attentive pooling scheme, which employs both images and ID features in generating attention weights. DICM also utilizes the visual connections between user preferences and ads, improving the performance remarkably.

To summarize, our contributions are three folds:



**Figure 1: (a) Typical display ad on App of Taobao. (b) Display advertising pipeline**

First, we propose the novel AMS framework. It goes beyond the well-known parameter distributing style with a sub-model distributing style, and facilitates the jointly learning of the whole model in a distributed manner. This is an important step towards enabling deep learning model to exploit large-scale and structured data with affordable computation and storage resources.

Second, we propose the DICM. It not only models ad with its image, but also exploits user's massive behavior images to better model user preference, which is much more challenging than that only uses ad images. We show that either ad images or user behavior images can benefit CTR prediction, and their combination will bring further significant improvement.

Moreover, we validate the efficacy and efficiency of our approach with extensive offline and online experiments. It has been now deployed in Taobao's display advertising system, serving the main traffic for half a billion users and millions of advertisers.

## 2 RELATED WORK

Early CTR prediction focuses on carefully designed low-dimensional statistical features, normally defined by votes of users' clicks, *etc.* [1, 28]. LS-PLM [8], FTRL [20] and FM [22] are classical explorations on shallow models. Recently, along with the number of samples and the dimension of features become larger and larger, CTR models evolve from shallow to deep. Especially, inspired by natural language processing field, the embedding technique which learns distributed representations is used to handle large scale sparse data. NCF [12] and Wide&Deep [5] exploit MLP network to greatly enhance model capacities. DeepFM [9] further models feature interactions by updating the wide part with factorization machine in Wide&Deep. The latest work DIN [31] proposes to apply attentive mechanism to adaptively model user behaviors according to a given item. These work have advanced the employment of sparse features. However, IDs only tell objects are different, and reveal little semantic information. Especially when an ID occurs in low frequency in training data, its parameters will not be well trained. And unseen ID during training will not take effect in prediction. Images with visual semantic information would bring better generalization ability of models. Further, unseen images in training data can still help CTR prediction with a well trained image model.

Image representation task makes a significant improvement in recent years. High level semantic features learnt by deep models [11, 16, 24, 26] have been proved to be effective on a large range of tasks. Some previous works try to introduce image information

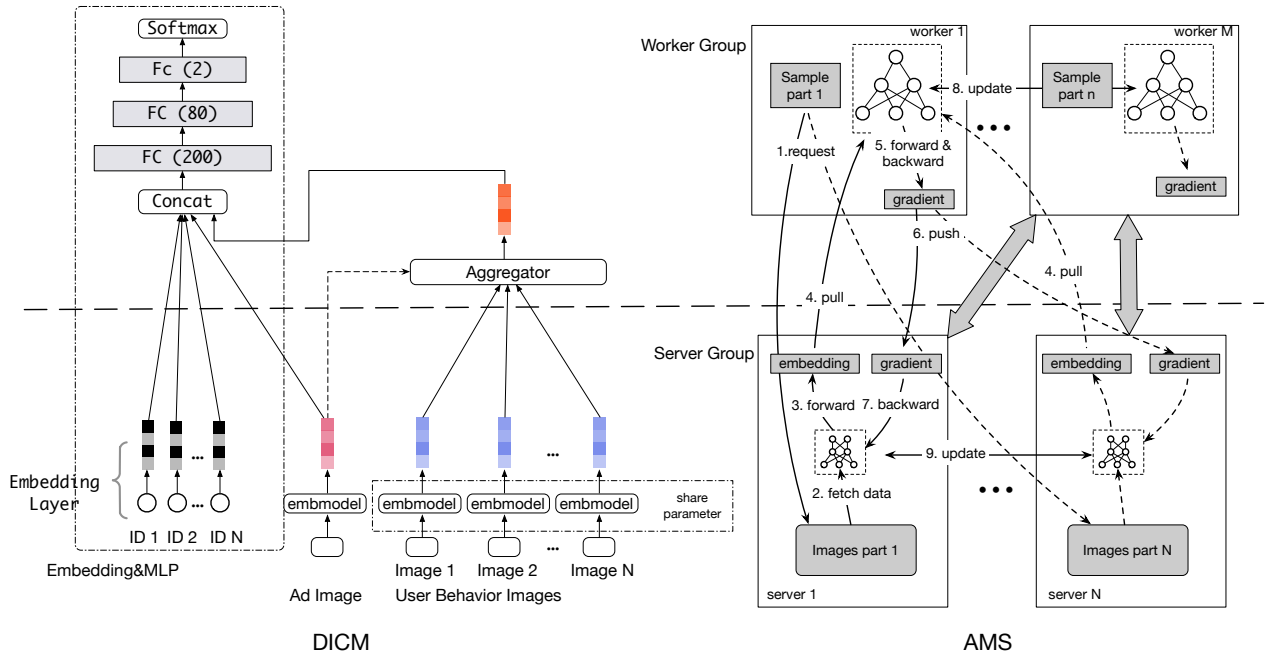


Figure 2: DICM network architecture implemented by Advance Model Server

in CTR model to describe ads. Cheng et al. [4] and Mo et al. [21] address the cold start problem by modeling ad images with either manually designed feature or pre-trained CNN models. Lynch et al. [19] introduce the visual information of items to overcome the misunderstanding of text-only representation in Esty's search engine. Chen et al. [3] propose to train the CNN in an end-to-end manner. All these works focus on representing ads with images, which differs from our motivation. Images of ads describe visual features of ads, and user behavior images will reveal visual preferences of users. Combining them together and bridging these visual information would result in better performance than either of them alone. In this paper, we propose to enhance the user representation with images and design a novel and efficient distributed machine learning paradigm to tackle the challenges brought by it.

### 3 DEEP IMAGE CTR MODEL

#### 3.1 Display advertising system

Taobao's display advertising system responds to billions of page view (PV) requests everyday. For each request, the most suitable ad is shown to a specific user in a specific scenario (viewing time, advertising position, etc.). The advertising system chooses the one ranked highest under eCPM mechanism from tens of millions of ads within only tens of milliseconds.

The online system completes this task in a funnel-like way and roughly consists of three sequential modules, as shown in Fig. 1(b). The *matching* module retrieves roughly around 4k ads from all candidates by current user's preferences inferred from its behaviors. The successive *Pre-rank* module further shrinks the number of candidates to about 400 with a light-weight CTR model. Finally,

the *Rank* module predicts CTR of ads accurately with sophisticated models and ranks them by eCPM to give the best choice. All of these modules depend on the appropriate understanding of user interests to give personalized recommendations. Fig. 1(a) shows a typical advertising result in Taobao mobile app.

In this paper, we focus on better user modeling with user behavior images in CTR prediction. In the following sections, we carefully describe the challenges and solutions by taking *Rank* as an example. Such method is straightly extended to DSSM-like [13] architecture for *Pre-rank*, which is omitted in this paper due to space limitation. Our approach can also be used with tree based deep model [33] in *Matching*, and we leave this as future work.

#### 3.2 Problem Formulation

Being fed features about user, ad and scenario *etc.*, CTR model outputs a probability that the ad will be clicked by the user in that scenario. Following previous works [3, 6, 21, 31], it is considered as a binary classification problem, whose label is the weak feedback—being clicked or not. And the cross-entropy error [7] is used as the objective function during training.

Different from classifying images which are well represented by pixels, CTR prediction problem needs carefully feature design with respect to the specific application. The common practice is to describe various aspects of user, item and scenario in each sample with underlying IDs, constituting many sparse feature fields. User historical behavior field consists of the ids of items the user previously clicked and is the most important one describing users. Such method leads to large scale but extremely sparse data.

Embedding with MLP fashion networks [5, 9, 12] are now widely used to fit such large sparse inputs. In Taobao's advertising system, highly optimized CTR models following this pattern are deployed. Embedding&MLP part of Fig. 2 shows a simplified version of the production model for clarity. Recently DIN [31] is introduced in production to better model sparse behavior features. Working with these sophisticated models, in the following sections we show that modeling user behavior with images can still bring significant improvements.

### 3.3 Modeling with image

We extend Embedding&MLP model with visual information, especially enhance the user behavior representations with images. We refer to this structure as Deep Image CTR Model (DICM) and refer to Embedding&MLP as the basic net. As illustrated in Fig. 2, users' behavior images and ad images are incorporated as two special feature fields. These images are first fed through a trainable sub-model to get high level representations with low dimensionality. Similar to embedding, the sub-model is also a kind of embedding operation which embeds image to vector. So we call it embedding model. The embedding model can be regarded as a generalizable extension of traditional key-value embedding operation, for it can embed new images unseen during training. Since user behaviors are of variable quantity, multiple embedded images need to be aggregated together into a fixed-length user representation, and then be fed into MLP.

It is worth noting that image embedding in this model is actually independent, *i.e.*, it does not depend on other features. Thus the embedding model can be forward/backward separately. This observation prompts us to design the Advanced Model Server. Moreover, more embedding models for various types of data, *e.g.* text, videos, can be devised using AMS.

## 4 ADVANCED MODEL SERVER

The main challenge of training is the huge quantity of images involved in user behavior. The image not only is large-size datasource itself, but also needs complex computation in extracting semantic information. For CTR prediction, each sample contains a user description with its massive historical behaviors. Therefore, the training system inevitably faces the heavy load of storage, computation and communication. For example, a typical user would have over 200 behaviors during our statistic period, which means a single training sample will involve over 200 images, hundreds of times more than that merely employs ad image. Moreover the training system needs to handle billions of training samples and finish model update daily, which is required for online production.

Advanced Model Server (AMS) provides an efficient distributed training paradigm by introducing a shared and learnable sub-model on servers which significantly reduces the size of features to be transmitted. AMS goes beyond the classical Parameter Server [17, 25] in the sense that AMS is of capable of forwarding and updating this sub-model shared by all server nodes.

### 4.1 From Parameter Server to AMS

Parameter Server (PS) is a widely adopted distributed architecture for large scale parametric machine learning problem. It consists of two node groups: the *worker group* and the *server group*. The *worker*

---

#### Algorithm 1 Advanced Model Server

---

##### Task Scheduler:

- 1: Initialize worker models  $\mathcal{W}$  and embedding models  $\mathcal{E}$
- 2: **for** Train with mini-batch  $t = 0, \dots, T$  **do**
- 3:   do WORKERITERATION( $t$ ) on all workers.
- 4: **end for**

##### Worker: $r = 1, \dots, M$

- 5: **function** WORKERITERATION( $t$ )
- 6:   load training data of mini-batch  $t$ :  $X_r^t, Y_r^t$
- 7:   request SERVEREMBED with image indices in  $X_r^t$
- 8:   pull all embeddings  $e_r^t$  from servers
- 9:   forward and backward  $\mathcal{W}$  with  $e_r^t$
- 10:   gradient w.r.t worker param  $\delta_{\mathcal{W}_r}^t = \nabla_{\mathcal{W}} \ell(X_r^t, Y_r^t, e_r^t)$
- 11:   gradient w.r.t embeddings  $\delta_{e_r}^t = \nabla_e \ell(X_r^t, Y_r^t, e_r^t)$
- 12:   push  $\delta_{e_r}^t$  to servers' SERVERUPDATE
- 13:   synchronize  $\delta_{\mathcal{W}_r}^t$  with all workers and update  $\mathcal{W}$

##### 12: **end function**

##### Server: $s = 1, \dots, N$

- 13: **function** SERVEREMBED( $t$ )
  - 14:   get image data  $I$  from local
  - 15:   compute embeddings  $e = \mathcal{E}(I)$
  - 16: **end function**
  - 17: **function** SERVERUPDATE( $t$ )
  - 18:   compute gradients w.r.t embedding model at server  $s$
  - 19:    $\delta_{\mathcal{E}_s}^t = \nabla_{\mathcal{E}} \ell(I) \cdot \delta_e^t$
  - 20:   synchronize  $\delta_{\mathcal{E}_s}^t$  with all servers and update  $\mathcal{E}$
  - 21: **end function**
- 

*group* contains a set of workers which do training on its assigned part of training samples. Meanwhile, the *server group* serves as a distributed database which stores parameters of the models and can be accessed through key-value structure. In this way, PS store and maintain a large parameter set in server without redundancy.

The Embedding&MLP model can be efficiently implemented with the PS-like architecture on GPU cluster. The parameters of embedding layer are placed in *server group* since their size far exceeds the memory capacity of each worker, and can be accessed (forward) and updated (backward) through key-value structure.

However, when image features, especially the tremendous user behavior related images, are employed, to complete the training procedure is not trivial.

Actually if images are stored in *worker group* along with training samples, then image features will greatly increase the training data size (in our scenario, from 134M Bytes to 5.1G per mini-batch, about 40 times larger), which makes it unaffordable for IO or storage.

And since the image feature is high-dimensional (typically 4096-D in our experiments) and far beyond that of ID features (typically 12-D), if images are stored in *server group* and accessed by workers during training, then it will bring heavy communication pressure.

To handle this issue, the key motivation of AMS is that by learning a unified embedding model in servers, the high dimensional raw image features can be embedded into low dimensions before being transmitted to workers.

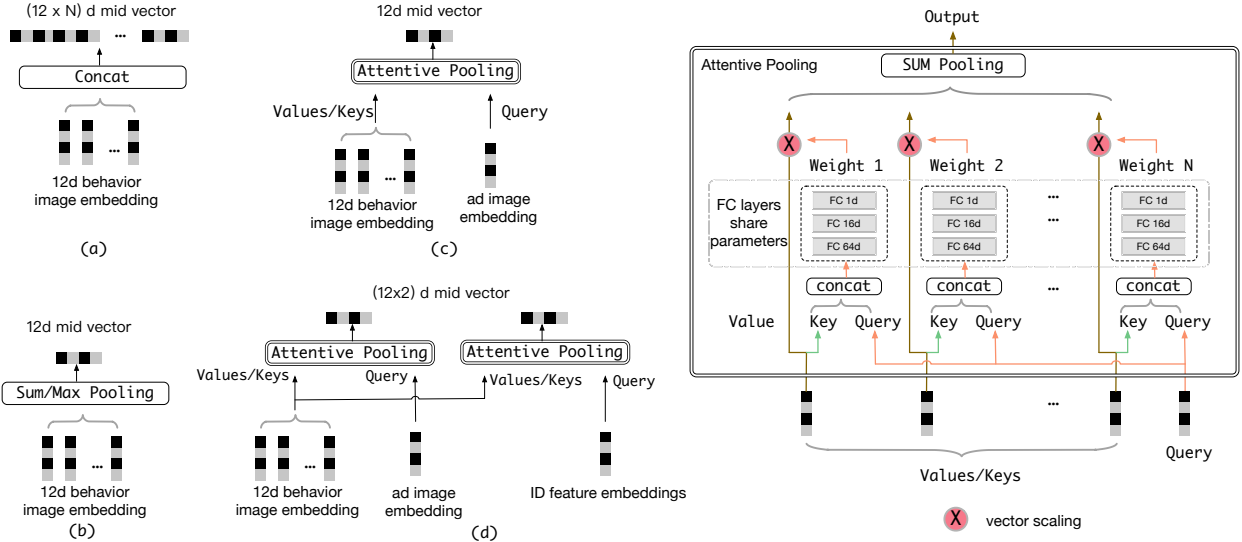


Figure 3: Aggregator architectures. (a) Concatenate (b) Sum/Max Pooling (c) Attentive Pooling (d) MultiQuery-AttentivePooling

## 4.2 AMS architecture

In this section, we detail the architecture of AMS, as illustrated in Fig. 2 and Algorithm 1. AMS includes two parts of nodes, i.e. servers and workers. Training samples are distributed among all workers and only image indices are contained in the samples for image features. Image data are evenly distributed among all servers without duplicates in key-value format, in which the key is an image index and the value is its image data.

In each iteration, each worker node first reads a mini-batch of samples separately and then requests the image data with image indices occurred in the samples from server nodes. Note that image data are evenly distributed on servers, requests from different workers for the same image will be gathered by the same server and processed only once. When receiving the request, the server first fetches the image data from local memory and then feeds it through the embedding model  $\mathcal{E}$  to get an embedded vector  $e$ . Then each worker pulls back embedded vector  $e$  instead of raw image data from servers to complete worker model computation and obtain gradients  $\delta_{W_r}$  and  $\delta_{e_r}$  for worker model and embeddings  $e$  respectively. Once again,  $\delta_{e_r}$  for the same image are collected by the same server and they are then back propagated through the embedding model to compute the gradient  $\delta_{\mathcal{E}_s}$  w.r.t embedding model at server  $s$ . Finally workers and servers synchronize and accumulate their model gradients  $\delta_{W_r}$  and  $\delta_{\mathcal{E}_s}$  and finish the model update.

Our AMS should be distinguished from PS. By the design of PS, parameters are distributed stored in *server group*, and servers are mutually independent without communication. While in AMS, image data are distributed stored and parameters of embedding model are globally shared. In other words, the parametric model in servers is essentially unified and is distributed trained by all servers. The gradient synchronization among servers in each training iteration is critical in AMS. Note that although PS supports user defined functions in servers [17], these functions are fixed and are not trainable.

AMS brings several benefits. First, the storage of images is significantly reduced by only storing once in servers. Further the communication is reduced for the embedding vectors are much smaller than original data (typically from 4096-D to 12-D, over 340 times compression ratio). Another benefit is that the computation of a certain image occurred several times within one training iteration can be naturally merged by servers, which reduces the computation load. It is also worth noting that servers and workers are actually deployed in the same GPU machines physically, so the alternative worker and server computation maximizes the GPU usage.

## 4.3 DICM implemented by AMS

As shown in Fig. 2, DICM can be efficiently trained with AMS. The embeddings of sparse ID features and embedding model are running in servers as design. MLP and Aggregator (detailed in following section) are running in workers.

The distributed GPU training architecture equipped with AMS makes daily updating model with tens of days log data come true, which is critical for real advertising system. Table 1 depicts the training time of our best configured model with 18 days of data with different number of GPU. It is noted that our system has desirable nearly linear scalability with GPUs. We use 20-GPU for a reasonable trade-off between efficiency and economy.

#GPU	5	10	20	40
Time(h)	62.9	32.0	17.4	10.2

Table 1: Training time with different number of GPU

## 4.4 Inference and online deployment

The efficiency is crucial for online deployment of the CTR model in large industrial advertising system. For CTR models with sparse ID features, e.g. Embedding&MLP, ID embeddings are globally placed in key-value storage. And parameters of MLP part are stored locally in a ranking server. For each request, the ranking server pulls the

ID embeddings and feed them through MLP to obtain the predicted CTR. This scheme is proved to be of high throughput and low latency in the production environment.

When involving images, especially large number of behavior images, extracting image features could bring heavy computation and communication load. The embedding model is separable from basic net. Benefitting from that the image embeddings can be computed in advance. So ranking servers are able to predict DICM efficiently with little modification. Note that the newly involved images can be embedded by the embedding model, which alleviates the cold start problem of ID features. DICM increases the response time over the baseline just in a tolerable degree, from 21 milliseconds to 24 milliseconds for each PV request.

## 5 IMAGE BASED USER MODELING

### 5.1 Image embedding model

The embedding model is designed to extract the pixel-level visual information to semantic embedded vector. Recent progress in computer vision shows that the learnt semantic features for classification tasks have good generalization ability [10, 24]. Our empirical studies show that VGG16 [24] performs better than trivial end-to-end training from scratch in our application. But due to the unsatisfactory complexity of VGG16, we adopt a hybrid training: the whole net is split into a fixed part followed by a trainable part that is end-to-end trained with CTR model.

For the fixed part, we adopt the first 14 layers of pre-trained VGG16 net [24], specifically, from Conv1 to FC6, which generates a 4096-D vector. It is a careful trade-off between efficacy and efficiency for practical application. E.g., replacing FC6 with 4096-D output by VGG16 FC8 with 1000-D output as the fixed part will lead to 3% relative performance loss in our experiments. It indicates that the information reduction in fixed part needs to be controlled, and the input size and the trainable part jointly learnt with the whole network are crucial. However, when we use lower layer in VGG16 as the fixed part, the computation load in training becomes high and we found the improvement is not significant. Finally VGG16 FC6 with 4096-D output is selected as the fixed part. For the trainable part, a 3-layer fully connected net (4096-256-64-12) is used and outputs a 12-D vector.

### 5.2 User behavior image aggregator

For CTR prediction with Embedding&MLP model, a compact representation of the user is critical. We need to aggregate the various user data, especially historical behaviors of variable number, to a fixed length vector. Thus an aggregator block is designed to aggregate numerous behavior image embeddings for this sake.

In fact, similar tasks are involved in many classical problems. For traditional image retrieval/classification, local features, e.g. SIFT [18], in an image are aggregated. Classical methods including VLAD [14] and sparse coding [29] accomplish this with sum or max operation. For neural machine translation, the context vector for different length sentence is abstracted with recent attentive method [2, 27]. We follow these ideas and explore various designs, especially the attentive method. Further ID feature information is concerned to propose the Multiple Query Attentive Pooling.

The most straightforward method is to **concatenate** all behavior image embeddings together, and pad or truncate to a specified length. But it would suffer a loss when the behavior number is large or when the behavior order changes. **Max** and **sum pooling** are another two direct methods, which can not focus appropriately for diverse user behaviors. Recently DIN [31] introduces attentive mechanism to user modeling. It adaptively captures the most relevant behaviors depending on the ad under consideration. We also employ this method, and considering the visual relevance, we use ad image as query in attention. We call this method **Attentive-Pooling**. These methods are illustrated in Fig. 3.

Interactions between different types of features are important. For example, the category id "T shirt" of ad and the "T shirt" image in user behaviors can be connected, and hence it captures the user's preference to such items better. Therefore, we propose the **MultiQueryAttentivePooling** (Fig. 3d) which incorporates both images and IDs for attentive weights generation. In detail, we design two attentive channels which involve ad image feature and ID feature as queries respectively. Both of the attentive channels generate their own weights as well as the weighted sum vectors separately, which are then concatenated. Note that different from the multi-head technique [27], the MultiQueryAttentivePooling uses different queries for each attentive channel and thus explores distinct relevances with complementarity.

We empirically compare these aggregator designs in Sec. 6.4.

## 6 EXPERIMENTS

### 6.1 Dataset and evaluation metric

The experiment data comes from Taobao's display advertising system. In detail, we construct a closed dataset with log data from arbitrary 19 consecutive days in July 2017. We use the data of the first 18 days as training set and that of the subsequent day as test set. In total, the dataset has 3.9 billion training samples and 219 million test samples. All offline experiments are conducted with this dataset. We use 27 categories of ID features including user profiles, user behaviors, ad description and scenario description, which are simplified from the highly optimized online configuration. The unique user number is about 0.13 billion. About 1% ads are newly uploaded or updated every day.

For offline evaluation metric, we adopt AUC (Area Under ROC Curve) which is commonly used in advertising/recommender system [3, 5, 21]. Besides, we also use the Group AUC (GAUC) introduced in [31, 32].

GAUC is a weighted average of AUC over all users. The GAUC is formulated as:

$$GAUC = \frac{\sum_i \#impression_i * AUC_i}{\sum_i \#impression_i}$$

where  $\#impression_i$  and  $AUC_i$  are the number of impression and AUC corresponding to the  $i$ -th user, respectively. In real advertising system, GAUC is proved to be more effective to measure the performance than AUC or Cross Entropy Loss[32], since the system is personalized and focuses on prediction for each user.



## 6.2 Training details

To speed up training and reduce storage cost, we follow the common-feature technique [3, 8, 31]. In detail, we put together samples corresponding to identical users to form sample groups which share user relevant features as common-features, following [8].

To describe user behavior, we select the *click* behavior of one specific user in the past 14 days. Since raw data from real system is noisy, we select typical click behavior with reasonable long visiting elapse time. We empirically find such filtering strategy achieves better performance. The average user's behaviors are filtered from more than 200 to 32.6.

We use PReLU [10] as the activation for each layer since we empirically find its superiority. Adam [15] is adopted as parameter optimizer, with the learning rate initialized to 0.001 and decayed by 0.9 after each 24,000 sample batches. The model converges after 2 epochs (128K iterations in our scenario).

**Partial warm-up.** Parameter initialization is widely used. Benefitting from the daily update scheme of our system, we can use the trained model of last day as initialization without any extra cost. It is observed that each part of DICM converges at different speed. The ID embedding is prone to get overfitting due to the sparsity of ID and the large parameter size, while image embedding model requires sufficient training to capture the highly non-linear relation between visual information and user's intention. So we propose the *partial warm-up* technique. In detail, we use pre-trained (but with training data of different date) model as initialization of all parts except ID embedding (*i.e.* image embedding model, extractor and MLP part), and randomly initialize the ID embedding part.

## 6.3 Efficiency study of AMS

We first study the efficiency superiority of AMS in our application. In detail, we compare with the following two possible ways to save the involved images as follows:

- **store-in-worker.** Storing the images in the worker nodes, along with other training data.
- **store-in-server.** Storing the images in the server nodes as the global dataset and fetching them in key-value format (without sub-model in server).

To give the quantitative results, we summarize our typical scenario. There are in total 3.9 Billion training samples processed by a 20-node GPU cluster. For each training iteration, the mini-batch is set to 3000 per node thus the effective minibatch size is 60000. In each sample, the user is related to 32.6 behavior images on average. Benefitting from common-feature technique (*c.f.* 6.2), each effective minibatch involves about 320k images as well as 1.4 million IDs (excluding images' IDs) according to statistic. There are in total 0.12 billion unique images involved in training, and each one is pre-processed to 4096-D float feature as training input.

We compare the AMS with the two replacements in Table 2. We see the AMS achieves nice system efficiency, while the *store-in-worker* and *store-in-server* strategies suffers major weakness *w.r.t.* storage or communication load. In detail, *store-in-worker* requires 31 times larger storage than that of AMS (5.1G *v.s.* 164M); while *store-in-server* costs 32 times more communication than that of AMS (5.1G *v.s.* 158M).

Strategy	Storage		Communication	
	Worker	Server	All	Image
store-in-worker	5.1G(332T)	0	128M	0
store-in-server	134M(8.8T)	30.3M(2T)	5.1G	5.0G
AMS	134M(8.8T)	30.3M(2T)	158M	30M

**Table 2: Efficiency study of AMS. "Storage" denotes the storage requirements of worker or server group for saving both image and ID data. "Communication" denotes the communication load between worker and server nodes to communicate all data (denoted by "All") and only image data (denoted by "Image"). The listed numbers are the average data size (in Bytes) for each mini-batch of the whole cluster, and these in brackets are the summation of the whole training.**

## 6.4 Ablation studies

We first individually study various design details of our approach with offline experiments in this section. For fair comparison, partial warm-up strategy is disabled for all ablation studies unless otherwise specified.

**Baseline.** We set our baseline model for all offline experiments as the Embedding&MLP model with only sparse ID features, as shown in Fig. 2, which is a simplified version of the production model in Taobao's display advertising system for clarity. Note that two special ID fields are also employed as sparse features in baseline: the IDs of ad image and the IDs of user behavior images. These two ID fields are essential for a fair comparison, because image features in fact can play a partial role of IDs and we should keep a common basis for both models to show a clean improvement of image semantic information. Besides, we adopt adaptive regularization [31] to tackle the overfitting problem of ID features.

**Study on image information.** DICM integrates both user behavior images and ad images. In this section, we conduct an ablation study of the effectiveness of them. To this end, we start from the baseline and separately use ad images, behavior images and both of them. Table 3 depicts the results on the offline dataset. It is observed that either behavior images or ad image will boost the baseline, showing the positive effect by introducing visual feature in user and ad modeling. Further, jointly modeling with both behavior images and ad image will significantly improve the performance. It is worth noting that the joint gain is much larger than the sum of gains brought by them individually, *i.e.*, 0.0055 *v.s.* 0.0044 in GAUC and 0.0037 *v.s.* 0.0024 in AUC. This result strongly indicates the cooperative effect of modeling users and ads by visual information, a desirable effect brought by our DICM.

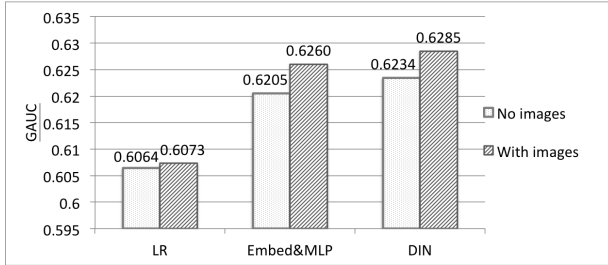
**Study on behavior image aggregator.** We detail the effects of different aggregators described in Sec. 5.2 with which the behavior image embeddings are exploited in the model. The results are shown in Table 4. The observations are 3 folds: i) Concatenation is not appropriate for behavior aggregation, providing inferior performance; sum/max pooling give reasonable improvements. ii) AttentivePooling shows a remarkably gain with ad images as attention queries. iii)

Method	GAUC	GAUC gain	AUC	AUC gain
baseline	0.6205	-	0.6758	-
ad image	0.6235	0.0030	0.6772	0.0014
behavior images	0.6219	0.0014	0.6768	0.0010
<b>joint</b>	<b>0.6260</b>	<b>0.0055</b>	<b>0.6795</b>	<b>0.0037</b>

**Table 3: Comparison of behavior images and ad image, and their combination in DICM.**

Aggregator	GAUC
baseline	0.6205
Only ad images	0.6235
Concatenation	0.6232
MaxPooling	0.6236
SumPooling	0.6248
AttentivePooling	0.6257
<b>MultiQueryAttentivePooling</b>	<b>0.6260</b>

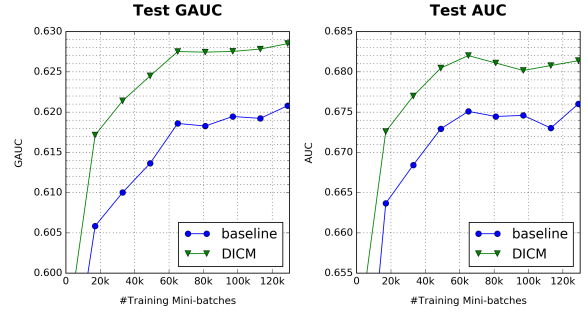
**Table 4: Result of different aggregator. Aggregators are investigated jointly with ad image.**



**Figure 4: GAUC of model with different basic structures**

MultiQueryAttentivePooling brings best results, benefitting from interactions between sparse ID and semantic information in images.

**Study on different basic structure.** Our work focuses on enhancing CTR prediction model with jointly involving visual information of user behaviors and ads. The basic network structure design for traditional sparse features is not the central topic in this paper. We assume that DICM can apply to different basic networks and bring consistent improvement with image features. To verify it, we test DICM with the classical Logistic Regression (LR) model and recently proposed DIN [31] model, along with the baseline Embedding&MLP as basic model. Fig. 4 compares offline metric GAUCs of these models. It can be seen that models with images consistently outperform their counterpart with only ID features as expected. DIN with image features performs the best, and largely surpasses classical DIN. The improvement of LR when enhancing with images is not as much as others. It is because LR can not fully utilize the high level semantic information of images.



**Figure 5: Offline result**

**Study on partial warm-up.** We empirically study warm-up strategies by comparing the strategy of non warm-up, partial warm-up and full warm-up. As shown in Table 5, partial warm-up performs best. Full warm-up leads to the worse result due to a severe overfitting in the ID embedding parameters.

warm-up strategy	Non	<b>Partial</b>	Full
GAUC	0.6260	<b>0.6283</b>	0.6230

**Table 5: Comparison of warm-up strategy**

## 6.5 Results of DICM

In this section, we compare the best configured DICM using partial warm-up strategy and MultiQueryAttentivePooling with the baseline by offline metric. Online A/B test is also conducted and shows a significant improvement over the state-of-the-art in production.

**Offline results.** We first evaluate our DICM model with offline dataset. Partial warm-up strategy and MultiQueryAttentivePooling are adopted. Table 6 and Fig. 5 show the AUC/GAUC comparison between baseline and the best configured DICM. DICM outperforms the baseline by 0.0078 GAUC and 0.0055 AUC, which actually are significant improvements in the real system. Moreover, it is noted from Fig. 5 that the gap between baseline and DICM is consistent during the training process, which indicates the robustness of our approach.

Method	GAUC	AUC
baseline	0.6205( $\pm 0.0002$ )	0.6758( $\pm 0.0003$ )
<b>DICM</b>	<b>0.6283(<math>\pm 0.0002</math>)</b>	<b>0.6814(<math>\pm 0.0003</math>)</b>
Absolute gain	0.0078	0.0055

**Table 6: Offline result, averaged by 5 runs**

**Online A/B test.** In online A/B test, we set the baseline as the production model, which is the state-of-the-art of our production environment. The production model is extended from the basic net by adding sophisticated cross-product features, such as user age  $\times$  ad category, user gender  $\times$  weekday *etc.* To be fair, we extend



DICM with the same features to get the online version of DICM. The comparison is conducted between the online version of DICM and the production model. Three key metrics of advertising system are considered: the CTR, eCPM and gross merchandise value per mile (GPM). As listed in Table 7, the DICM achieves a consistent gain in the online A/B test during a 7-day long statistic period. And these improvements are also remarkable because they show that DICM brings 9.2% more product impressions and 5.9% more sales for advertisers and 5.7% more revenue for the platform. Considering Taobao's massive volume of traffic and merchandise, the commercial value of DICM is significant. DICM has been now deployed in Taobao's display advertising system, serving the main traffic for half a billion users and millions of advertisers.

Date	CTR	eCPM	GPM
Day1	+10.0%	+5.5%	+3.3%
Day2	+10.0%	+6.8%	+8.0%
Day3	+9.1%	+6.6%	+1.8%
Day4	+9.9%	+4.8%	+7.9%
Day5	+8.2%	+5.0%	+2.7%
Day6	+8.2%	+5.4%	+9.9%
Day7	+9.0%	+5.7%	+8.0%
Average	9.2( $\pm 0.7$ )%	5.7( $\pm 0.7$ )%	5.9( $\pm 4.0$ )%

**Table 7: Relative increments of DICM in online A/B test, counting in 7 consecutive days (Nov 21st-27th, 2017)**

## 7 CONCLUSION

In this paper, we have proposed a novel and efficient distributed machine learning paradigm called AMS. Benefitting from it, we manage to utilize massive number of behavior images in capturing user interest for CTR prediction in display advertising. We design a complete architecture named DICM that jointly learns ID and visual information for both user and ad description, and depict its superiority via offline and online experiments. Since user behaviors usually incorporate abundant cross-media information such as comment texts, detailed descriptions, images and videos, we believe that our proposed AMS and the model study can also benefit future work in this direction.

## REFERENCES

- [1] Deepak Agarwal, Bee-Chung Chen, and Pradheep Elango. 2009. Spatio-temporal models for estimating click-through rate. In *Proceedings of the 18th international conference on World wide web*. ACM, 21–30.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
- [3] Junxuan Chen, Baigui Sun, Hao Li, Hongtao Lu, and Xian-Sheng Hua. 2016. Deep ctr prediction in display advertising. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 777–785.
- [4] Haibin Cheng, Roelof van Zwol, Javad Azimi, et al. 2012. Multimedia features for click prediction of new ads in display advertising. In *Proceedings of the 18th ACM SIGKDD*. ACM, 777–785.
- [5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
- [6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
- [7] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. A tutorial on the cross-entropy method. *Annals of operations research* 134, 1 (2005), 19–67.
- [8] Kun Gai, Xiaoqiang Zhu, Han Li, Kai Liu, and Zhe Wang. 2017. Learning Piecewise Linear Models from Large Scale Data for Ad Click Prediction. *arXiv preprint arXiv:1704.05194* (2017).
- [9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. *arXiv preprint arXiv:1703.04247* (2017).
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*. 1026–1034.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
- [12] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 173–182.
- [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 *Proceedings of the 22nd ACM CIKM*. ACM, 2333–2338.
- [14] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. 2010. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 3304–3311.
- [15] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- [16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.
- [17] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*, Vol. 1. 3.
- [18] David G Lowe. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Vol. 2. Ieee, 1150–1157.
- [19] Corey Lynch, Kamelia Aryafar, and Josh Attenberg. 2016. Images don't lie: Transferring deep visual semantic features to large-scale multimodal learning to rank. In *Proceedings of the 22nd ACM SIGKDD*. ACM, 541–548.
- [20] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, et al. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD*. ACM, 1222–1230.
- [21] Kaixiang Mo, Bo Liu, Lei Xiao, Yong Li, and Jie Jiang. 2015. Image Feature Learning for Cold Start Problem in Display Advertising. In *IJCAI*. 3728–3734.
- [22] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.
- [23] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. ACM, 521–530.
- [24] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
- [25] Alexander Smola and Shrawan Narayanamurthy. 2010. An architecture for parallel topic models. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 703–710.
- [26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, et al. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.
- [27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *CoRR abs/1706.03762* (2017). [arXiv:1706.03762](https://arxiv.org/abs/1706.03762)
- [28] Xuerui Wang, Wei Li, Ying Cui, Ruofei Zhang, and Jianchang Mao. 2010. Click-through rate estimation for rare events in online advertising. *Online Multimedia Advertising: Techniques and Technologies* (2010), 1–12.
- [29] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. 2009. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. IEEE Conference on*. IEEE, 1794–1801.
- [30] Shuai Zhang, Lina Yao, and Aixun Sun. 2017. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435* (2017).
- [31] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. 2017. Deep Interest Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1706.06978* (2017).
- [32] Han Zhu, Junqi Jin, Chang Tan, Fei Pan, Yifan Zeng, Han Li, and Kun Gai. 2017. Optimized Cost per Click in Taobao Display Advertising. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2191–2200.
- [33] Han Zhu, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. 2018. Learning Tree-based Deep Model for Recommender Systems. *arXiv preprint arXiv:1801.02294* (2018).