Image Matters: Jointly Train Advertising CTR Model with Image Representation of Ad and User Behavior

Tiezheng Ge, Liqin Zhao, Guorui Zhou, Keyu Chen, Shuying Liu
Huiming 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}@alibaba-inc.com, jingshi.gk@taobao.com

***Abstract*—Click Through Rate(CTR) prediction is vital for online advertising system. Recently sparse ID features are widely adopted in the industry. While the ID features, *e.g.* the serial number of ad, are of low computation complexity and cheap to acquire, they can reveal little intrinsic information about the ad itself. In this work, we propose a novel Deep Image CTR Model(DICM). DICM i) introduces image content features to exploit the intrinsic description of ad/goods and shows effectiveness on complete dataset in accordance with the product environment of a real commercial advertising system; ii) not only represents ad with image features, but also, for the first time, jointly models the user behaviors and ads with image features to capture user interests. However, the users historical behaviors involve massive images for industry scale training(*e.g.* 2.4 million images per mini-batch with the batchsize of 60k), which brings great challenges on both computation and storage. To tackle the challenges, we carefully design a highly efficient distributed system which enables daily-updated model training on billions of samples essential for product deployment. Extensive experiments show that the image features can be effective representations as well as good complements to the corresponding ID features.**

I. Introduction

Display advertising is a typical form of advertisement for Internet corporations such as Google, Alibaba and Tencent. Cost-per-click(CPC) is a widely adopted payment method for display advertising. In CPC mode, the ad publishers rank the candidate advertisements by eCPM(effective cost per mille)), which can be estimated by multiplying the bid price(a known value) by the (estimated) click through rate(CTR). Such strategy makes CTR prediction the core task in the advertising system.

Traditional researches on CTR prediction largely focus on carefully designed feedback feature [1, 2] and shallow models(*e.g.*, Logistic Regression [3]). Recently, benefiting from the rapid development of both computing/storage power and machine learning algorithm, the CTR prediction system has evolved in two aspects: i) large scale sparse ID feature(either one-hot or multi-hot encoded) is used instead of the feedback feature, and ii) multi-layer neural networks [4, 5] are used to replace shallow models. Compared with feedback features, the sparse ID feature(*e.g.*, goods ID, user ID) can represent user and candidate ads more accurately. However, such IDs, take goods ID for example, contains no descriptive information about the goods it refers to, and on the contrary, the information is generated from the co-occurrence of samples.

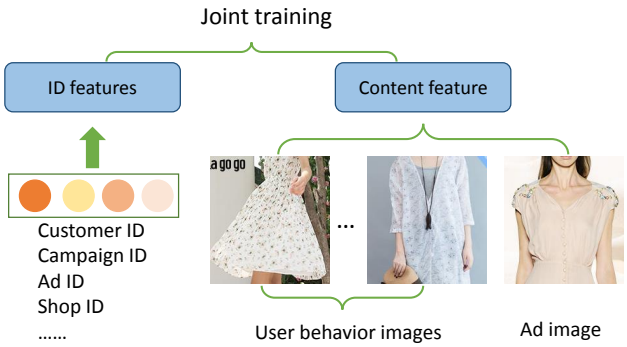


Fig. 1: Joint training using both ID features and content features.

Consequently, the models based on large scale sparse IDs usually do well in *memorization* rather than *generalization*. For instance, the ID-based models are prone to get stuck in predicting CTR of new goods with unobserved IDs, known as the cold start problem.

In this paper, we try to update CTR model based on *ID features* to that based on both *ID features* and *content feature*, as illustrated in Fig. 1. Totally different from the ID feature which only tell whether two entities are identical, content feature can express their similarity. For example, to describe an ad, we can i) just assign it a unique ID, that is ID feature, or ii) describe the raw-pixel or other representation of its corresponding image, that is content feature. We believe the content feature is more informative and has better generalization ability. Take the content feature of ad—its image feature—for example, ads that have visually similar images are more likely to share similar CTR. Such kind of relation can hardly be encoded by original ID feature.

Among various content features such as images, texts and videos, our method focuses on one important kind—the image feature. We are motivated by the typical scenario when a user is shown an image ad on a website and he/she needs to determine whether or not to click the ad. Since the user can only get information from the image, the real problem is to predict the CTR of *images* rather than the *ad*. In fact, the user cannot distinguish ads that share the same display images and these ads are certainly to share the same CTR. Meanwhile, the same ads displayed with different images perform differently

on CTR. Therefore, we start from the image feature, studying the significance of content features for CTR prediction.

In fact, there are some pioneering works (*cf.* [6, 7]) dedicating to explore the positive effects of image features in CTR prediction. They mainly focus on the image of ads. However, as we know, joint modeling of the user and the advertisement is the core task in current CTR prediction system. What we concern most is whether a particular user will click on a particular ad. On a website, user historical behaviors are logged in the system and most of them are interactions with displayed goods images, like click or purchase. Can we simultaneously involve image information in modeling both the ads and the users and obtain a synergy effect? The answer is yes.

In this paper, we propose a framework, named Deep Image CTR Model(DICM), that jointly models the advertisement and the user with image content. Specifically, we describe one user with the images of goods involved in the user's historical behaviors(clicking, purchasing, favoring, *etc.*), and represent the ad with the ad image. The intuition comes from that if one user clicks on an image, then he/she is more likely to be interested in goods with visually similar images. We adopt the popular deep neural network, define CTR prediction as a classification problem, and get significant improvements in the real advertising system.

We carefully design the network architecture. To efficiently and effectively extract image features, the image extractor of DICM is divided into *fixed part* whose parameters are fixed and *trainable part* whose parameters are updated during training. To adaptively model user behavior with image features, we make use of attentive pooling mechanism with context aware attentive weights [8] which is proved to outperform straightforward sumpooling/maxpooling.

For an industrial application, the introduction of user behavior images will inevitably bring great challenges. On one hand, the involved images will cost a large amount of storage. On the other hand, the extraction of image features relies on a highly non-linear transformation, which requires huge computation. Take an actual training system as an example: Each mini-batch of training data (batch size = 60k) involves up to 2.4 million images of user behavior, while the whole image set of each training task is about 200 million, that is, at TB magnitude. Besides, over 10 layers of neural networks are employed to process the input image.

In order to handle the computation and storage pressure caused by massive image data, we design an efficient training system. In detail, we design a distributed GPU training system with multi(typically 20) workers, which utilizes the data parallelism and model parallelism. Ultimately, our training process with billions of sample is managed to complete within 24 hours, which means our online model can be daily updated.

The contributions of our work are summarized as follows:

1) We take *image feature* as a typical kind of *content feature* and propose to jointly model ad and user with image content. To the best of our knowledge, this is the first system to co-train ad images and user images.
2) We carefully design proper model structure to extract image features and propose to use attentive pooling to adaptively model user behavior with a fixed length vector.
3) We build a highly efficient model training system to handle massive training samples and images from a real advertising platform.

II. Related Work

Our work is closely related to two kinds of techniques—CTR prediction in advertising/recommendation system and image representation in computer vision system, both of which have been developed independently for a long time.

As aforementioned, the CTR prediction system actually evolves in two aspects. The first one is the progress on input features. Early research on CTR prediction lies on carefully designed low-dimensional statistic features, normally defined by votes of users' clicks, *etc.* [1], as well as their variant, *e.g.*, the one with smoothing technique [2]. In recent years, large amounts of works try to build model based on high-dimensional sparse IDs feature(such as ad IDs or user IDs) to achieve more accurate description [5, 4]. The second one is the development of prediction model, from classical Logistical Regression [9] and other shallow model such as decision trees [10] to the well known multi-layer neural networks [5, 4].

On the other hand, image representation, the core task of computer vision community, has attracted tremendous attention. Traditional work largely relies on handcraft feature extraction, among which GIST [11], HOG [12], *etc.* give global description, while others such as Visual Bag of Visual Word [13], VLAD [14], Fisher vector [15] choose the manners of local feature aggregation. However, recent progress on deep learning, especially CNN [16, 17, 18, 19], motivates us to directly explore the raw pixels by multi layer non linear mapping. A common practice is to adopt well trained networks(without head part) as feature extractor and transfer to related task(*e.g.*, [20, 21, 22])

Our work lies in the intersection of CTR prediction and image description. In fact, there are some previous work trying to introduce image information in ad description. Cheng et al. [23] addresses the cold start problem by adding manually designed feature generated from images of ads. Mo et al. [7] extracts feature from ad images by CNN models pre-trained in classification task. Chen et al. [6] proposes to involve images in an end-to-end network targeting on CTR prediction. These works only focus on the features generated from images of ads, but ignore that from images of goods appearing in the users' browsing history.

III. Deep Image CTR Model

A. Problem statement

Following previous works [24, 6, 7, 5], we consider the Click Through Rate(CTR) prediction as a two-class classification problem. A basic unit, usually called *impression*, is an incident that a specific ad is shown to a specific website user, in a specific scenario(viewing time, advertising position,

etc.). Our goal is to judge the possibility that the user would click this ad, in other words, to predict the output score of the click/non-click classification in the given impression. In the training phase, the binary label whether an impression is clicked is provided to generate objective function by cross-entropy error [25].

B. From ID features to content features

Our baseline CTR prediction model is built on (sparse) ID features. Taking well selected ID features as input, this model generates output as:

$$\mathbf{y} = \mathbb{F}(\mathbf{id_1}, \mathbf{id_2}, \dots, \mathbf{id_N}) \tag{1}$$

where each of $\mathbf{id_1}$, $\mathbf{id_2}$, ... , $\mathbf{id_N}$ denotes a category of ID feature representing an individual attribute of the given impression. It is either one-hot vector(*e.g.*, ad id, user id, displaying position id) or with multiple none-zero entry(*e.g*, user behavior id) depending on the its own property. $\mathbb{F}$ is the adopted mapping from input features to the 2-D click/non-click possibility vector, denoted by $\mathbf{y}$.

We learn from the standard format which divides $\mathbb{F}$ to 2 modules: the *representor* which separately extracts each category of ID feature to one *middle vector*, followed by the *discriminator* which concatenates all *middle vectors* to give final result. In formal, Eq. 1 turns to:

$$\mathbf{y} = \mathbb{F}_d(\mathbb{F}_r^1(\mathbf{id_1}), \mathbb{F}_r^2(\mathbf{id_2}), \dots, \mathbb{F}_r^N(\mathbf{id_N})) \tag{2}$$

where $\mathbb{F}_r^i$ is the representor corresponding to i-th ID feature and $\mathbb{F}_d$ is the succeeding discriminator.

Then, adding content feature is a natural extension of above formulation: we only need to involve the content feature vectors as additional input items and design proper representor, which can be formulated as:

$$\mathbf{y} = \mathbb{F}_d(\mathbb{F}_r^1(\mathbf{id_1}), \dots, \mathbb{F}_r^N(\mathbf{id_N}), \mathbb{F}_r^{N+1}(\mathbf{x_{N+1}}) \dots) \tag{3}$$

where $\mathbf{x_{N+1}} \dots$ are the newly introduced content features associated with representor $\mathbb{F}_r^{N+1} \dots$. $\mathbf{x_{N+1}}$ is either 1-D vector descriptor for single entry feature (*e.g.*, the image feature for ad description), or 2-D matrix for multiple entry feature (*e.g.*, the multiple images representing user behavior)

Recent proposed CTR prediction model which adopted ID features along with neural network [4, 5] strictly follow the formulation of Eq. 1. We borrow such framework as baseline, and extend it with content feature by Eq. 3.

C. Modeling with images

In this paper we propose to employ image, a vital and informative content feature, to simultaneously model advertizement and user behavior. However, it keeps uncertain which image to use in ad and user modeling.

For ad, it is intuitive and has been successfully adopted [7, 24, 23] that ad can be represented by the image viewed on the webpage as the entry of the ad. For user behavior, the representation image is not so straightforward, since there are various describable behaviors, such as purchasing, favoring and clicking. In this paper we study the *click* behavior, in the sense that *click* is the most general user response that is able to bring rich descriptions. Another issue is which image is related to *click*. In this paper, we follow the natural sense to use the image one user was viewing when he/she did *click* behavior.

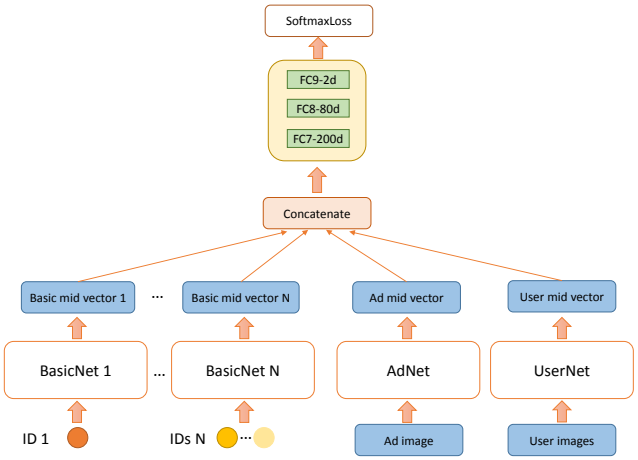


Fig. 2: DICM network architecture

In the rest part of this paper, we simply use *ad image* to denote the image representing the ad, and use *user images* to denote a set of images representing one's historical behavior.

IV. Network Design

The overall pipeline of our work is illustrated in Fig. 2. We follow state-of-the-art CTR prediction systems [4, 5] to build the baseline, which is divided to *representor* module and *discriminator* module. As shown in Fig. 2, the *representor* module consists of a set of sub-networks, called *Basicnet*, each of which takes one category of ID(s)(either one-hot or multi-hot) as input and outputs a fix-length *middle vector*. We follow [4] and [5] to simply design *Basicnet* as one-layer fully connected net, which can be implemented efficiently by ignoring non-visited parameters due to the sparsity [24]. The output width of *Basicnet* is a hyper parameter which is set to 12 in our pipeline. In the *discriminator* module, all extracted *middle vectors* from *Basicnet* are concatenated, processed by Batch Normalization [26] in order to remove scale differences and finally feed into a 3-layer fully connected net to get 2-class likelihood vectors.

Starting from the baseline model, we further add our content feature, specifically, the related images into system. According to Eq. 3, we incorporate both *ad image* and *user images* into *representor* module as individual input features. Following recent progress in image representation, we carefully design sub-networks to extract their *middle vector*, which is denoted by *Adnet* for *ad image* and *Usernet* for *user images*. In the next subsections, we will describe them separately.

A. Adnet

As illustrated in Fig. 3, the *Adnet* can be viewed as a feature *extractor* which takes raw pixels of (resized) image

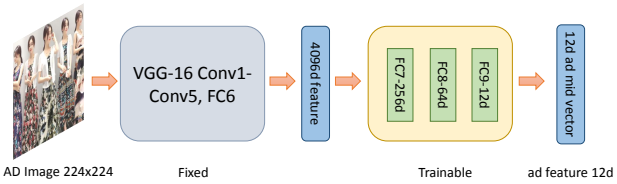


Fig. 3: Adnet architecture

as input and produces fixed length (12 in our work) output vector with multi-layer net. In related works, the *extractor* model is either pre-trained [7] or fully trained in end-to-end style [24]. We consider it as an efficiency-effectiveness trade off, and propose a hybrid version: we split the whole net into *fixed part* whose parameters is fixed and *trainable part* that is involved in training.

The design of *fixed part* is quite flexible and largely depends on the task. Here we select the first 14 layers of popular VGG16 net [17], specifically, from Conv1 to FC6. This net will generate a 4096-D vector as input of *trainable part*. For *trainable part*, we simply use a 3-layer fully connected net, which is quite fast and can achieve enough fitting ability for industrial application.

B. Usernet

As shown in Fig. 4, the *Usernet* contains two parts: the *extractor* and the *aggregator*. Similar to that in *Adnet*, the *extractor* in *Usernet*, which also has a *fixed part* and a *trainable part*, extracts fixed length vector feature from *user images* one by one. The *aggregator* is designed to pool all *user features* extracted from *user images* of particular user into one vector. Instead of straightforward dimension-wise sumpooling/maxpooling, we propose the attentive pooling mechanism which have shown superiority in user modeling [5] and neural machine translation [8]

The sumpooling can be viewed as equally weighted sumpooling, corresponding to the assumption that each user behavior share equal importance in every impression scenario. However, it is not always the case. Many e-commercial site users have quantity of behaviors with large diversity. When equally weighted summed, *user images* which are related to the *ad image* would be inevitably interfered by the unrelated *user images*. To alleviate this problem, we introduce the attention network to highlight relevant behaviors while depress the irrelevant ones. Specifically, the attention network uses concatenated *user features*(extracted), *ad middle vector* and other *basic middle vectors* as input and infers 1-D output with a 3-layer fully connected network. All inferred outputs, each of which associates one *user feature*, are normalized by a softmax layer to attentive weights. Finally, weighted sumpooling is performed according to the attentive weights.

V. System Implementation

The basic model (without images) is implemented on a distributed GPU training platform named X-Deep Learning(XDL), which is firstly introduced in [5]. To handle the

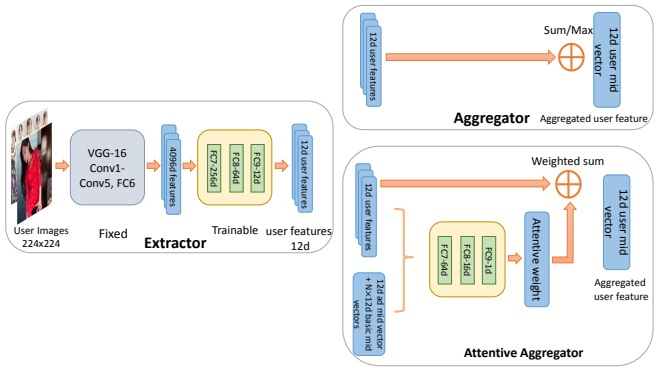


Fig. 4: Usernet architecture

tremendous images involved in the training as well as the extra process brought by *Adnet* and *Usernet*, we update the XDL to XDLv2. Fig 5 depicts pipeline of XDLv2.

As aforementioned, the *Basicnet* of is actually a full connect net, which can be efficiently implemented due to the sparse property. That is, to directly select the 12-D parameter vector of the FC parameter matrix corresponding to non zero entry if ID vector, and sum-pool all selected vectors. Following this intuition, XDL contains two parts: the *feature service* module and the *discrimination service* module. The *feature service* module, which holds a distributed parameter database of *Basicnet*, response parameter vector for each query of ID. Thus it provides model parallelism. Meanwhile, the *discrimination service* perform training phase individually in each work node by query ID and fetch parameter vector from *feature service*, which provides data parallelism. These two module are connected with communication component.

A. Advanced feature service

The main improvement of XDLv2 is the *advanced feature service*—the feature service with optional operation on site. As illustrated in Fig. 5, the *advanced feature service* distributedly keeps two datasets of images involved in the *Adnet* and *Usernet* respectively. We only save the *middle feature* extracted by the fixed part of *extractor*, since the fixed part would not be updated. In forward phrase of training, when receiving query image id, the *advanced feature service* fetches corresponding image feature from database and directly performs the *extractor* processing, and then sent the processed vector to *discrimination service* module. In backward, the *advanced feature service* receives the gradient from *model service* and update the parameter of *extractor* with back propagation.

In general, the *advanced feature service* can be viewed as a Parameter-Server(PS) [27] enhanced with more complicated feature extractors. There are two insights for this design: i) Communication load would be greatly reduced since the image feature is shrunk by extractor (typically from 4096-D to 12-D). ii) Queries from *different discrimination service* node can be naturally merged and share the extractor process, which greatly reduces computation load.

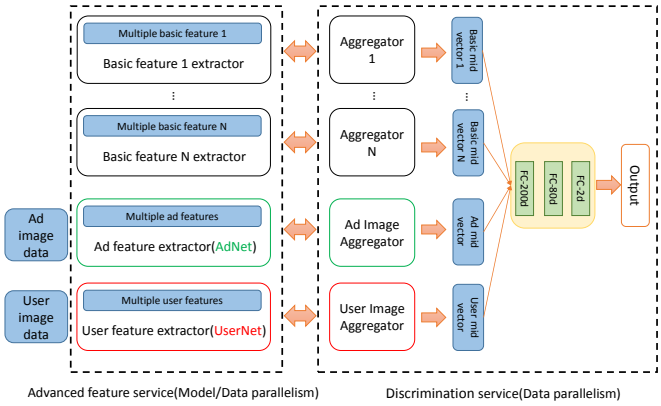


Fig. 5: System overview

VI. Training Detail

For baseline model, we follow the online platform configuration and use 27 categories of ID features including user profiles, user behavior, ad description and scenario description. Since we focus on the comparison of image content feature and the corresponding ID feature, we do not utilize extra tricks such as cross-product feature. To speed up training and reduce storage cost, we follow the common-feature technique [5, 6]. In detail, we gather samples corresponding to identical user to form sample groups which share user relevant feature as common-feature.

For training, we employ a 20-GPU cluster, on which the *advanced feature service* and *discrimination service* runs in a time-sharing way. To further speed up the time-consuming image fetching, we cache images with high frequency in memory, and design the asynchronous pre-fetch process. We set the batch size to 3,000 per parallel node, thus the overall batch size is 60,000. 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 pick *typical click behavior* according to the visiting elapsed time, visiting frequency, etc. On average, each user have 32.6 behaviors according to statistics, and there are altogether 200 million unique images involved in training task.

We use PReLU [28] as the activation for each layer since we empirically find its superiority. We use Adam [29] as parameter optimizer, with the learning rate initialized to 0.001 and decayed by 0.9 after each 24,000 sample batches.

A. Partial warm-up

Due to the different data representation, the difficulties of training *Basicnet*, *Adnet* and *Usernet* are totally different, thus make them hard to converge at the same speed. The *Adnet* and *Usernet* are relatively deep to extract highly non-linear image feature, which need intense training. Meanwhile, the *Basicnet* is quite shallow but with large quantity of parameters, thus is prone to get overfitting. Therefore, it is wise to design the learning scheme that update *Adnet* and *Usernet* more frequently than *Basicnet*. Here we propose the *partial warm-up*. In detail, we use pre-trained (but with training data of different date) model as initialization of *Adnet* and *Usernet* and randomly initialize the rest part of model. Benefit from the daily update scheme of our system, we can use the trained model of last day as initialization without any extra cost. We find it works well and does not need require any change in the training phase.

VII. Experiments

A. Dataset and evaluation metric

The experiment data comes from a real advertising system. In detail, we construct a close 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 last day as test set. The size of training set is 3.9 billion while that of test set is 219 million. The model converges after 2 epochs(128K iterations in our scenario).

Instead of commonly used AUC(Area Under ROC Curve), we adopt the *Group AUC*(GAUC) as offline evaluation metric, which is introduced in [30, 5]. GAUC can be viewed as a weighted average of AUC for each user. 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[30].

To make GAUC more sensitive when comparing with baseline, we propose *Relative GAUC*(R-GAUC), which is similar to *Relative AUC*[6, 31]:

$$R\text{-}GAUC = (\frac{GAUC(method) - 0.5}{GAUC(baseline) - 0.5} - 1) * 100\%$$

B. Main results

we set the following baseline and, for reference, list the result of the basic ID feature inferred by Logistic Regression(LR) model.

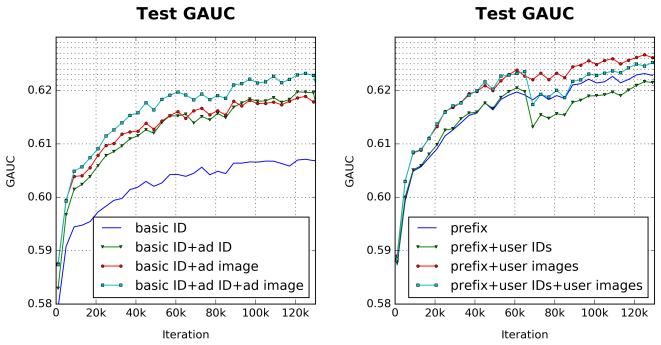
- *basic ID*—our baseline, which are ID features except *ad ID* and *user IDs* involved in our system. We use it as the baseline for R-GAUC calculation in the following experiments.
- *basic ID-LR*, the result of our basic ID inferred by widely used Logistic Regression(LR). Note that it is not baseline of this paper, for we are exploring the positive effect of image information rather than the model structure.

To evaluate the effectiveness of image features, we focus on two pair of ID features:

- *ad ID* and *ad image*, which give each ad an ID/image description.
- *user IDs* and *user images*, of which *user IDs* are the IDs of goods involved in specific user's *click* behavior and *user images* are the corresponding image descriptions of the goods. Note that the *user IDs* is NOT the ID of the specified user.

Method	GAUC	GAUC gain	R-GAUC
basic ID(baseline)	0.6082	0.0000	0.00%
basic ID-LR	0.5980	-0.0102	-9.43%
basic ID+			
-	0.6082	0.0000	0.00%
ad ID	0.6207	+0.0125	+11.55%
ad image	0.6195	+0.0113	+10.44%
ad ID+ad image	**0.6242**	**+0.0160**	**+14.79%**
basic ID+ad ID+ad image+			
-	0.6242	+0.0160	+14.79%
user IDs	0.6230	+0.0148	+13.68%
user images	**0.6281**	**+0.0199**	**+18.39%**
user IDs+user images	0.6263	+0.0181	+16.73%

TABLE I: image(s) *v.s.* ID(s)



(a) (b)

Fig. 6: Comparison on ad/user ID(s) and ad/user image(s) (*prefix* denotes basic ID+ad ID+ad image)

1) ad image v.s. *ad ID:* We first evaluate the effectiveness of *ad image*. We use model with *basic ID* as baseline, and successively add *ad ID*, *ad image* and the combination of them. The result is illustrated in Table I and Fig. 6(a). We observe large margin between baseline and that with *ad ID*. It is not surprising, for *ad ID* is the key ID feature which provides accurate description of ad. It is desirable to see that *ad image* significantly outperforms baseline, which indicates that the *image feature* provides highly descriptive and discriminative information about the items WITHOUT the help of ID. What's more, the combination model can bring notable gain over each single feature. Since the amount of parameters in the *Basicnets* is directly proportional to the number of *ad ID*, the *ad ID* owns a large number of parameters, thus do well in memorization. While *ad image* contains more descriptive information but much less parameters in *Adnet*, thus has better generalization ability.

2) user images v.s. *user IDs:* Then, we turn to verify the effectiveness of *user images*. We fix the ad description to be our best achieved version—combination of *ad ID* and *ad image*, and successively add *user IDs*, *user images* and their combination. As depicted in Table I and Fig. 6(b), we find that the *user IDs* in fact show negative effect, which is caused by severe overfitting problem(see Fig.7). In our scenario, the *user IDs* are 2 orders of magnitude larger than the *ad ID* (100 millions *v.s.* millions), thus bring the same orders of parameter increment and inevitably lead to overfitting. However, with

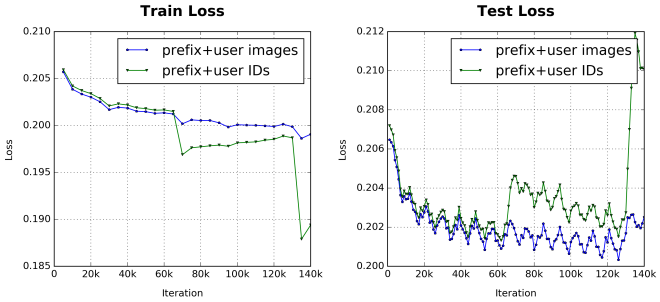


Fig. 7: Overfitting problem of *user IDs* (*prefix* denotes basic ID+ad ID+ad image)

much less trainable parameter, our model with *user images* can perfectly overcome this problem and further achieve better performance through simultaneously modeling ad and user with image content.

From Table I, we also observe the superiority of our DNN based model over traditional shallow model(basic ID *v.s.* basic ID), which is reported by previous literatures(*e.g.*, [6]).

C. Study on architecture and training

To handle the industrial scale training samples and newly involved image features, we carefully design our system and training scheme based on extensive experiments.

1) Study on image feature extractor: Since the *extractor* is the fundamental part of both *Adnet* and *Usernet*, to explore its proper structure is critical for the final result. As aforementioned, the *extractor* is divided into *fixed part* and *trainable part*. In this section we empirically study them separately. Since we are studying image net, for simplicity, we only test the model of *basic ID+ad image*(no warm-up).

We first focus on the *fixed part* by keeping the *trainable part* as default configuration. We consider the following candidates:

- VGG16 [17]: Our adopted net with 4096-D output, which is described above.
- VGG16_tuning: VGG16 networks fine tuned with images from our goods images labeled by its product category(*e.g.*, shoes, books ...).
- Resnet152 [19]: Popular 152-layer classification networks. We remove the last layer and utilize the rest subnet whose output is 2048-D.
- VGG16 + Resnet152: Concatenation of the output of VGG16 and Resnet152, with 6144-D in total.

Table II lists the GAUC result of the model with above *fixed part*. We observe comparable performance of these CNN-based methods (VGG16, VGG16_tuning and Resnet152), and little improvement of feature combination, which shows these homologous image features are not complementary. Hence, we use solely VGG16 as default configuration.

We have also tested different output of VGG16 by sliding the output layer from Conv5 to FC8. The results are appended to Table II. We note that with less fixed layers, the performance is better, but the computing and storage cost is much higher.

According to the results, VGG16 FC6 is a good trade-off between performance and cost.

Method(output width)	R-GAUC
VGG16 (4096-D)	+8.78%
VGG16_tuning (4096-D)	+8.69%
Resnet152 (2048-D)	+7.58%
VGG16 + Resnet152 (6144-D)	+8.96%
VGG16 FC6 (4096-D)	+8.78%
VGG16 Conv5(25088-D)	+11.00%
VGG16 FC7 (4096-D)	+8.13%
VGG16 FC8 (1000-D)	+5.73%

TABLE II: Comparison on *fixed part*, with *ad image*

We go ahead to the *trainable part*. This time, we heuristically change the layer numbers and output dimension. The result is shown in Table III. We find that the performance is not very sensitive to the structure change within proper range. So we safely adopt 3 Layers 12-D configuration.

Method (FC layer configuration)	R-GAUC
3 Layers 12Dim (256-64-12-D FC)	+8.78%
4 Layers 12Dim (256-64-32-12-D FC)	+8.69%
2 Layers 12Dim (256-12-D FC)	+8.60%
3 Layers 24Dim (256-64-24-D FC)	+8.69%

TABLE III: Comparison on *trainable part*, with *ad image*.

2) Study on training time: We test the training time of our best configured model(*basic ID* + *ad ID* + *ad image* + *user images*, 2 epochs) by setting GPU number as 5, 10, 20 and 40 respectively. The following table depicts the result. It is noted that our system has desirable nearly linear scalability with workers. 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

3) Study on pooling strategy: To evaluate the impact of pooling strategy of the *aggregator* of *Usernet*, we vary our best model, the *ad ID+ad image+user images*, by replacing the attentive pooling of the *aggregator* to maxpooling and sumpooling. As shown in table, attentive pooling performs best with considerable margin.

Pooling strategy	Attentive	Sum	Max
R-GAUC	**+18.39%**	+17.01%	+15.71%

4) Study on partial warm-up: We evaluate our *partial warm-up* technique. Again, we try our best model with/without warm up of *extrator* part of *Adnet* and *Usernet*. Note that the init model is trained on samples of different date. As shown in the following table, the warm-up version shows superiority without any additional system change and training cost, as the training process runs in periodic manner.

Method	With warm-up	Without warm-up
R-GAUC	**+18.39%**	+17.19%

Fig. 8: Visualizing joint training. The *ad image* is on the left-hand side, while the *user images* are on the right-hand side. The *user images* in the first row are with high attention weights and that in the second row are with low weights.

D. Visualizing joint training

We explore the cooperative effect of joint training on *ad image* and *user images* by case study. Fig. 8 depict two cases. We find that the *user images* visually similar to the *ad images* are assigned large weight, and vise versa. That is, the system automatically select representative behavior images to form user description depending on the *ad image*.

E. Online A/B test

As our method is developed upon industrial scenario, it is critical to verify its superiority in the practical advertising system. We conduct A/B test via live experiment, and report the relative CTR/eCPM gain benefiting from image information.

In detail, we randomly allocate part of the flow to our proposed strategy(with image) and the rest part to baseline strategy(without image). For fair comparison, we expand the *basic ID* to sophisticated feature sets used in our online baseline. We count the CTR and eCPM within 24 hours.

As shown in the following table, the image involving system outperforms baseline by considerable gain, which brings large revenue for Internet corporations.

Online index	CTR	eCPM
Relative gain with image	+7.1%	+4.4%

VIII. Conclusion

We have proposed a CTR predict framework on display advertising, which employs image in both ad and user behavior modeling. To model user behavior, we use the goods image corresponding to the *click* behavior of specific website user. To handle billion scale industrial data, we carefully design the

network architecture and the multi-GPU training system. We show the superiority of our framework via extensive experiments and qualitative visualization. We hope this work can inspire future work aiming for CTR prediction with content feature(such as text, image and video).

REFERENCES

[1] D. Agarwal, B.-C. Chen, and P. Elango, "Spatio-temporal models for estimating click-through rate," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 21–30.

[2] X. Wang, W. Li, Y. Cui, R. Zhang, and J. Mao, "Click-through rate estimation for rare events in online advertising," *Online Multimedia Advertising: Techniques and Technologies*, pp. 1–12, 2010.

[3] M. Richardson, E. Dominowska, and R. Ragno, "Predicting clicks: estimating the click-through rate for new ads," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 521–530.

[4] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 2016, pp. 7–10.

[5] G. Zhou, C. Song, X. Zhu, X. Ma, Y. Yan, X. Dai, H. Zhu, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," *arXiv preprint arXiv:1706.06978*, 2017.

[6] J. Chen, B. Sun, H. Li, H. Lu, and X.-S. Hua, "Deep ctr prediction in display advertising," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 811–820.

[7] K. Mo, B. Liu, L. Xiao, Y. Li, and J. Jiang, "Image feature learning for cold start problem in display advertising." in *IJCAI*, 2015, pp. 3728–3734.

[8] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[9] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin *et al.*, "Ad click prediction: a view from the trenches," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2013, pp. 1222–1230.

[10] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 2014, pp. 1–9.

[11] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[13] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *null*. IEEE, 2003, p. 1470.

[14] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3304–3311.

[15] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *Computer Vision–ECCV 2010*, pp. 143–156, 2010.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[18] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[21] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 447–456.

[22] L. Wang, Y. Qiao, and X. Tang, "Action recognition with trajectory-pooled deep-convolutional descriptors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4305–4314.

[23] H. Cheng, R. v. Zwol, J. Azimi, E. Manavoglu, R. Zhang, Y. Zhou, and V. Navalpakkam, "Multimedia features for click prediction of new ads in display advertising," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 777–785.

[24] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[25] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[27] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server." in *OSDI*, vol. 1, no. 10.4, 2014, p. 3.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[29] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] H. Zhu, J. Jin, C. Tan, F. Pan, Y. Zeng, H. Li, and K. Gai, "Optimized cost per click in taobao display advertising," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 2191–2200.

[31] L. Yan, W.-j. Li, G.-R. Xue, and D. Han, "Coupled group lasso for web-scale ctr prediction in display advertising," in *International Conference on Machine Learning*, 2014, pp. 802–810.