Privileged Features Distillation for E-Commerce Recommendations

Chen Xu* Quan Li* Junfeng Ge, Jinyang Gao, Xiaoyong Yang, Changhua Pei, Hanxiao Sun, and Wenwu Ou
Alibaba Group, Beijing, China
{chaos.xc,liquan.quanli,beili.gjf,jinyang.gjy,xiaoyong.yxy,changhua.pch,hansel.shx,santong.oww}@alibaba-inc.com

ABSTRACT

Features play an important role in most prediction tasks of e-commerce recommendations. To guarantee the consistence of off-line training and on-line serving, we usually utilize the same features that are both available. However, the consistence in turn neglects some discriminative features. For example, when estimating the conversion rate (CVR), i.e., the probability that a user would purchase the item after she has clicked it, features like dwell time on the item detailed page can be very informative. However, CVR prediction should be conducted for on-line ranking before the click happens. Thus we cannot get such post-event features during serving, although they can be recorded for off-line training.

Here we define the features that are discriminative but only available during training as the privileged features. Inspired by the distillation techniques which bridge the gap between training and inference, in this work, we propose privileged features distillation (PFD). We train two models, i.e., a student model that is the same as the original one and a teacher model that additionally utilizes the privileged features. Knowledge distilled from the more accurate teacher is transferred to the student, which helps to improve its prediction accuracy. During serving, only the student part is extracted and it relies on no privileged features. To our knowledge, this is the first work to fully exploit the potential of such features. To validate the effectiveness of PFD, we conduct experiments on two fundamental prediction tasks in Taobao recommendations, i.e., click-through rate (CTR) at coarse-grained ranking and CVR at fine-grained ranking. By distilling the interacted features that are prohibited during serving for CTR and the post-event features for CVR, we achieve significant improvements over both of the strong baselines. Besides, by addressing several issues of training PFD, we obtain comparable training speed as the baselines without any distillation.

CCS CONCEPTS

• **Information systems → Information retrieval**; • **Computing methodologies → Neural networks**.

*Both authors contributed equally to this work.

KEYWORDS

Privileged Features, Distillation, E-commerce Recommendations, CTR, CVR

ACM Reference Format:
Chen Xu* Quan Li* Junfeng Ge, Jinyang Gao, Xiaoyong Yang, Changhua Pei, Hanxiao Sun, and Wenwu Ou. 2018. Privileged Features Distillation for E-Commerce Recommendations. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

In recent years, deep neural networks (DNNs) [3, 4, 9, 19, 25, 37] have achieved very promising results in the prediction tasks of recommendations. However, most of these works focus on the model aspect. While there are limited works except [3, 4] paid attention to the feature aspect in input, which essentially determine the upper-bound of the model performance. In this work, we also focus on the feature aspect, especially the features in e-commerce recommendations.

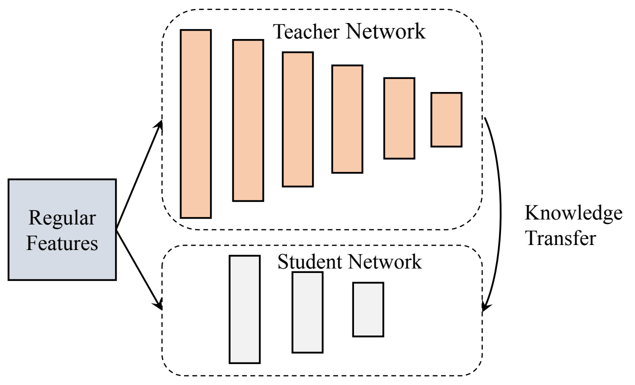
To ensure the consistence of off-line training and on-line serving, we usually use the same features that are both available in the two environments in real applications. However, a bunch of discriminative features, which are only available at training time, are thus abandoned. Taking conversation rate (CVR) estimation in e-commerce recommendations as an example. Here we aim to estimate the probability that the user would purchase the item if she clicked it. Features describing user behaviors in the clicked detail page, e.g., the dwell time on the whole page, whether viewing the comments or not, whether communicating with the seller of not, etc., could be very helpful in CVR estimation. However, these features cannot be utilized for on-line CVR prediction in recommendation, because it has to be done before any click happens. Although such post-event features can indeed be recorded for off-line training. In consistent with the learning using privileged information [30, 31], here we define the features that are discriminative for prediction tasks but only available at training time, as the *privileged features*.

A straightforward way to utilize the privileged features is multi-task learning, i.e., by predicting each feature with an additional task. However, in the multi-task learning, each task does not necessarily satisfy a no-harm guarantee (i.e. privileged features can harm the learning of the original model). More importantly, the no-harm guarantee will very likely be violated since estimating the privileged features might be even more challenging than the original problem [18]. From the practical point of view, when using dozens of privileged features at once, it would be very challenge to tune all of the tasks.

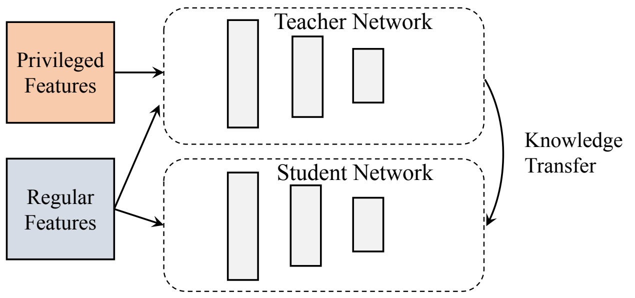
Inspired by the privileged information distillation technique [22], here we propose privileged features distillation (**PFD**) to take advantage of such features. We train two models, i.e., a student and a teacher model. The student model is the same as the original one, which processes the only features that are both available for off-line training and on-line inference. The teacher model processes all of the features, which include the privileged ones. Knowledge distilled from the teacher, i.e., the soft labels in this work, is then used to supervise the training of the student in addition to the original hard labels, i.e., $\{0, 1\}$, which additionally improves its performance. During on-line serving, only the student part is extracted, which relies on no privileged features as the input and guarantees the consistence with training. In PFD, the privileged features are combined in a more appropriate way for the prediction task. Generally, adding more privileged features will lead to more accurate prediction. Besides, PFD only introduces one extra distillation loss no matter what the number of privileged features is, which is much easier to balance.

PFD is different from the commonly used model distillation (MD) [2, 11]. In MD, both the teacher and the student are processing the same inputs. And the teacher uses models with more capacity than the student. For example, the teachers can use deeper networks to instruct the shallower students [17, 36]. Whereas in PFD, the teacher and the student are using the same models but differ in the inputs. We give an illustration on the difference in Figure 1. In this work, we are to apply PFD in Taobao recommendations. We conduct experiments on two fundamental prediction tasks by utilizing the corresponding privileged features. The contributions of this work are summarized as follows:

- We identify the privileged features existing in e-commerce recommendations. And we propose PFD to utilize them. As far as we know, this is the first work of fully exploiting the potential of such features, which are usually neglected in current recommendation systems.
- By applying PFD, we improve the performance of the original model, i.e., the student in the distillation framework, while not disturbing its inference during serving. Different from the widely used MD by distilling knowledge from more complex models, we are utilizing the much less explored privileged information distillation [22], e.g., by distilling from the privileged features described above. We find that the two distillation techniques are complementary, and could be combined to acheive further improvements.
- Under the huge-scale industry data, it could take very long time for the cumbersome DNN model to converge. Thus it is impractical to adopt distillation technique until the teacher has converged as traditionally does. Here we instead train the teacher and the student synchronously [1, 35, 36]. To stablize the training, we propose an adaptive update scheme, see, i.e., Algorithm 1. Besides, we share the embeddings for regular features that are both processed by the teacher and the student. After these modifications, PFD can reach comparable training speed as the original one without any distillation techniques, meanwhile giving much better results.
- We conduct experiments on two fundamental prediction tasks at Taobao recommendations, i.e., CTR prediction at coarse-grained ranking and CVR prediction at fine-grained ranking. By distilling the interacted features that are prohibited due to efficiency requirement for CTR at coarse-grained ranking and the post-event features for CVR as introduced above, we are able to make significant improvements over the strong baselines used currenty in Taobao.



(a) Model Distillation



(b) Privileged Features Distillation

Figure 1: Illustration of model distillation (MD)[11] and privileged features distillation (PFD) proposed in this work. In MD, the knowledge is distilled from the more complex model. While in PFD, we are distilling the knowledge from the privileged features.

2 RELATED DISTILLATION TECHNIQUES

Before giving detailed description of our PFD, we will firstly introduce the distillation techniques [2, 11]. Overall, the techniques are to help the non-convex student models to train better. For model distillation, we can typically write the objective function as follows:

$$\min_{\boldsymbol{W}_s} L_s\left(\boldsymbol{y}, f_s(\boldsymbol{X}; \boldsymbol{W}_s)\right) + \lambda * L_d\left(f_t(\boldsymbol{X}; \boldsymbol{W}_t), f_s(\boldsymbol{X}; \boldsymbol{W}_s)\right), \quad (1)$$

where f_t and f_s are the teacher model and the student model, respectively. L_s denotes the student pure loss with the known hard labels $\boldsymbol{y}$ and L_d denotes its loss with the soft labels produced by the teacher. $\lambda \in \mathbb{R}^+$ is the hyper-parameter to balance the two losses. Compared with the original function that minimizes L_s alone, we are expecting that the additional loss L_d in Eq.(1) will help to train $\boldsymbol{W}_s$ better by distilling the knowledge from the teacher. In the work of [26], Pereyra et. al. regard the distillation loss as regularization on the student model. When training f_s alone by minimizing L_s, it is prone to get overconfident prediction, which overfits the training set [28]. By adding the distillation loss, f_s will also approximate the soft prediction from f_t. By softening the outputs, f_s is more likely to achieve better generalization performance.

Typically, the teacher model is more powerful than the student model. Teachers can be the emsembles of several models [2, 11, 35], or DNNs with more neurons [29], more layers [17, 36], or even broader numerical precisions [24] than students. There are also some exceptions, e.g., in the work of [1], both of the two models are using the same structure and learned from each other, with difference only in the initialization and the orders to process the training data.

As indicated in Eq.(1), the parameter $\boldsymbol{W}_t$ of the teacher is fixed across the minimization. We can generally divide the distillation technique into two steps: firstly train the teacher with the known labels $\boldsymbol{y}$, then train the student by minimizing Eq.(1). In some applications, the models could take rather long time to converge, thus it is impractical to wait for the teacher to be ready as Eq.(1). Instead, some works try to train the teacher and the student synchronously [1, 35, 36]. Besides distilling from the final output as Eq.(1), it is possible to distill from the middle layer, e.g., Romero et al. [27] try to distill the intermediate feature maps, which help to train a deeper and thinner network.

In addition to distilling knowledge from more complex models, Lopez-Paz et al. [22] propose to distill knowledge from privileged information $\boldsymbol{X}^*$,

$$\min_{\boldsymbol{W}_s} L_s\left(\boldsymbol{y}, f(\boldsymbol{X}; \boldsymbol{W}_s)\right) + \lambda * L_d\left(f(\boldsymbol{X}^*; \boldsymbol{W}_t), f(\boldsymbol{X}; \boldsymbol{W}_s)\right), \quad (2)$$

Privileged information distillation is proposed to utilize $\boldsymbol{X}^*$ that is only available at training time. In the work of [8], Garcia et. al. extends the technique to action recognition, where they learn representations from depth and RGB videos, while relying on RGB data only at test time. Although being promising, privileged information distillation is much less explored in real applications. In this work, we further extend it to the prediction tasks in recommendation.

3 PRIVILEGED FEATURES IN TAOBAO RECOMMENDATIONS

To have better understanding of the privileged features exploited in this work, we firstly give an overview of Taobao recommendations in Figure 2. As usually done in industry recommendations [4, 21], we adopt the cascaded learning framework. There are overall three stages to select/rank the items before presenting to the user, i.e., candidate generation, coarse-grained ranking, and fine-grained ranking. To make a trade-off between efficiency and accuracy, more complex and effective model is adopted as the cascaded stage goes forward, while with the expense of higher latency to scoring the items. In the candidate generation stage, we choose around 10^5 items that are most likely to be clicked or purchased by one user from the huge scale corpus. Generally, the candidate generation is mixed from several sources, i.e., collaborative filtering [7], the DNN models [4], etc. After the candidate generation, we adopt two stages for ranking, where the PFD is applied in this work.

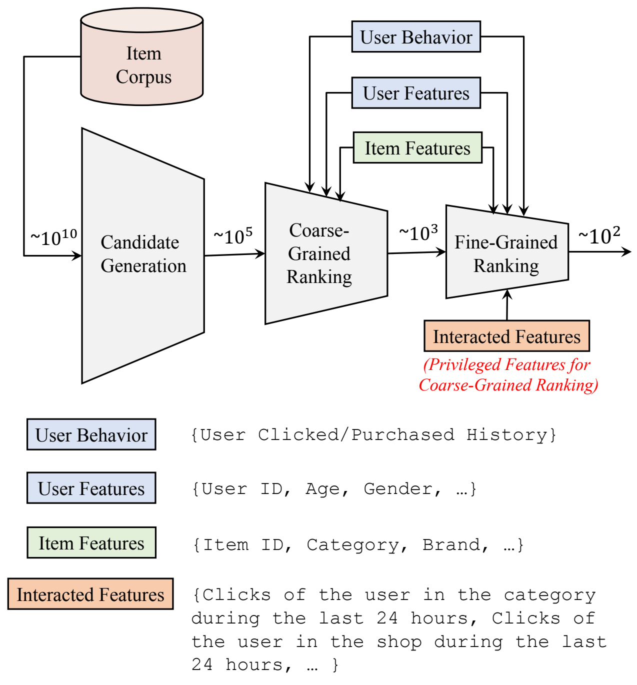

Item Corpus
~10^10
Candidate Generation
~10^5
Coarse-Grained Ranking
~10^3
Fine-Grained Ranking
~10^2
User Behavior
User Features
Item Features
Interacted Features
(Privileged Features for Coarse-Grained Ranking)
User Behavior {User Clicked/Purchased History}
User Features {User ID, Age, Gender, …}
Item Features {Item ID, Category, Brand, …}
Interacted Features {Clicks of the user in the category during the last 24 hours, Clicks of the user in the shop during the last 24 hours, … }

Figure 2: Overview of Taobao recommendations. Here we adopt a cascaded learning framework to select/rank items before presenting to users. At coarse-grained ranking, the interacted features, although being rather discriminative, are prohibited as they greatly increase the latency at serving. Some representative features are illustrated in the lower part of the figure.

In the coarse-grained ranking stage, we are mainly to estimate the CTRs of all items selected by the candidate generation stage, which are then used to select the top-k highest ranked items for the next stage. The inputs of the prediction model mainly consist of three parts. The first part is the user behavior, which records the history of her clicked/purchased items. As the user behavior is in sequential, RNNs [10, 12] or self-attention[16, 32] is usually adopted to model the user's long short-term interests. The second part is the user features, which contain user id, age, gender, etc. Across this work, all features are in one-hot encodings and we learn an embedding for each one[1]. We then concatenate the projected embeddings of all features into a long vector. The third part is the item features, which contain item id, category, brand, etc. Feature processing in this part also follows the same as the user ones.

At coarse-grained ranking stage, the complexity of the prediction model is strictly restricted, in order to grade tens of thousands of candidates in milliseconds. Here we utilize the inner product model [14] to measure the item scores:

$$f\left(\boldsymbol{X}^u, \boldsymbol{X}^i; \boldsymbol{W}^u, \boldsymbol{W}^i\right) \triangleq \left\langle \boldsymbol{\Phi}_{\boldsymbol{W}^u}\left(\boldsymbol{X}^u\right), \boldsymbol{\Phi}_{\boldsymbol{W}^i}\left(\boldsymbol{X}^i\right)\right\rangle, \quad (3)$$

where the superscript u and i denote the user and item, respectively. $\boldsymbol{X}^u$ denotes a combination of user behavior and user features.

[1]Numerical features are discretized with pre-defined boundaries.

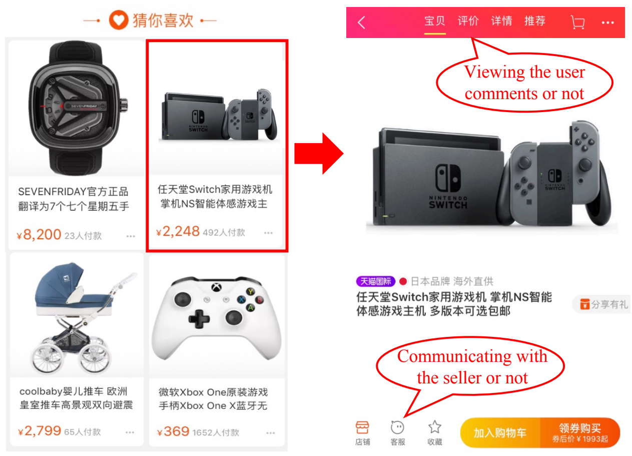

猜你喜欢
SEVENFRIDAY官方正品 翻译为7个七个星期五手
¥8,200 23人付款
任天堂Switch家用游戏机 掌机NS智能体感游戏主
¥2,248 492人付款
coolbaby婴儿推车 欧洲 皇室推车高景观双向避震
¥2,799 65人付款
微软Xbox One原装游戏 手柄Xbox One X蓝牙无
¥369 1652人付款
宝贝 评价 详情 推荐
Viewing the user comments or not
天猫国际 日本品牌 海外直供
任天堂Switch家用游戏机 掌机NS智能 体感游戏主机 多版本可选包邮
分享有礼
Communicating with the seller or not
店铺 客服 收藏
加入购物车
领券购买 券后价 ¥1993起

Figure 3: Illustrative features describing the user behavior in the detailed page of the clicked item. Including the dwell time that is not shown, these features are rather informative for CVR estimation. However, during serving, we need to use CVR to rank all candidate items as shown in left sub-figure before any item being clicked. We thus denote these features as the privileged features for CVR estimation.

$\Phi_{\boldsymbol{W}}(\cdot)$ represents the non-linear mapping with learned parameter $\boldsymbol{W}$. $\langle\cdot,\cdot\rangle$ is the inner product operation. As the user side and the item side are separated in Eq.(3), during serving, we can compute the mappings $\Phi_{\boldsymbol{W}^i}(\cdot)$ of all items off-line in advance[2]. When a request comes, we only need to execute *one* forward pass to get the user mapping $\Phi_{\boldsymbol{W}^u}(\boldsymbol{X}^u)$ and compute its inner product with all candidates, which is extremely efficient. For more details, see the illustration in Figure 4.

As shown in Figure 2, the coarse-grained ranking does not utilize any interacted features, e.g., clicks of the user in the item category during the last 24 hours, clicks of the user in the item shop during the last 24 hours, etc. As verified by the experiment below, adding these features can largely enhance the prediction performance. However, it in turn greatly increases the latency during serving. The interacted features are depending on the user and the specific item. In other words, the features vary with different items or users. If putting them either at the item or the user side of Eq.(3), the inference of the mappings $\Phi_{\boldsymbol{W}}(\cdot)$ need to be executed as many times as the number of candidates, i.e., 10^5 here. Generally, the non-linear mapping $\Phi_{\boldsymbol{W}}(\cdot)$ costs several orders more computation than the simple inner product operation. It is thus unpractical to use the interacted features during serving. Here we regard them as the *privileged features* for CTR estimation at coarse-grained ranking.

In the fine-grained ranking stage, besides estimating the CTR as done in the coarse-grained ranking, we will estimate the CVR for all candidates, i.e., the probability that the user would purchase the item if she clicked it. In the e-commerce recommendation, the main aim is to maximize the Gross Merchandise Volume (GMV), which can usually be decomposed as CTR × CVR × Price. Once getting the CTR and CVR for all items, we can then rank them by the expected GMVs. By the definition of CVR, it is obvious that user behaviors on the detailed page of the clicked item would be rather helpful for the prediction. We can extract several features describing the behavior, e.g., the dwell time on the whole detailed page, whether the user views the comment or not, whether the user communicates with the seller or not, etc. For better illustration, we give an example in Figure 3. The left sub-figure is main page with candidate items ranked by expect GMVs. And the right sub-figure is an example of detailed page after clicking the item. We also give an illustration of some features that are non-trivial for the purchase prediction in the detailed page. However, during serving, we need to estimate CVR for ranking before any future click happens. The features describing the user behavior on the clicked pages are not available. Although they can be recorded for off-line training. Here we denote these features as the *privileged features* for CVR estimation.

[2]In order to capture the real-time user preference, e.g., clicking on new items, the user mappings are not stored.

Algorithm 1 Minimizing the student, distillation, and teacher loss as Eq.(5) synchronously with adaptive λ

```
Input: $\lambda_0$, $\lambda_{inc}$, $\lambda_{max}$, boundaries $\{i_0, i_1, \ldots, i_n\}$, and learning
    rate $\eta$
 1: Initialize $(\boldsymbol{W}_s, \boldsymbol{W}_t)$
 2: Let $i = 0$ and $\lambda = \lambda_0$.
 3: while not converged do
 4:     Get training data $(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{X}^*)$.
 5:     Updating $(\boldsymbol{W}_s, \boldsymbol{W}_t)$ as follows:
        $\boldsymbol{W}_s = \boldsymbol{W}_s - \eta\nabla_{\boldsymbol{W}_s}\{L_s + \lambda * L_d\}$,
        $\boldsymbol{W}_t = \boldsymbol{W}_t - \eta\nabla_{\boldsymbol{W}_t}L_t$. //No distillation loss $L_d$
 6:     if $i \in \{i_0, i_1, \ldots, i_n\}$ then
 7:         Updating $\lambda = \min(\lambda + \lambda_{inc}, \lambda_{max})$.
 8:     end if
 9:     Updating $i = i + 1$
10: end while
Output: $(\boldsymbol{W}_s, \boldsymbol{W}_t)$
```

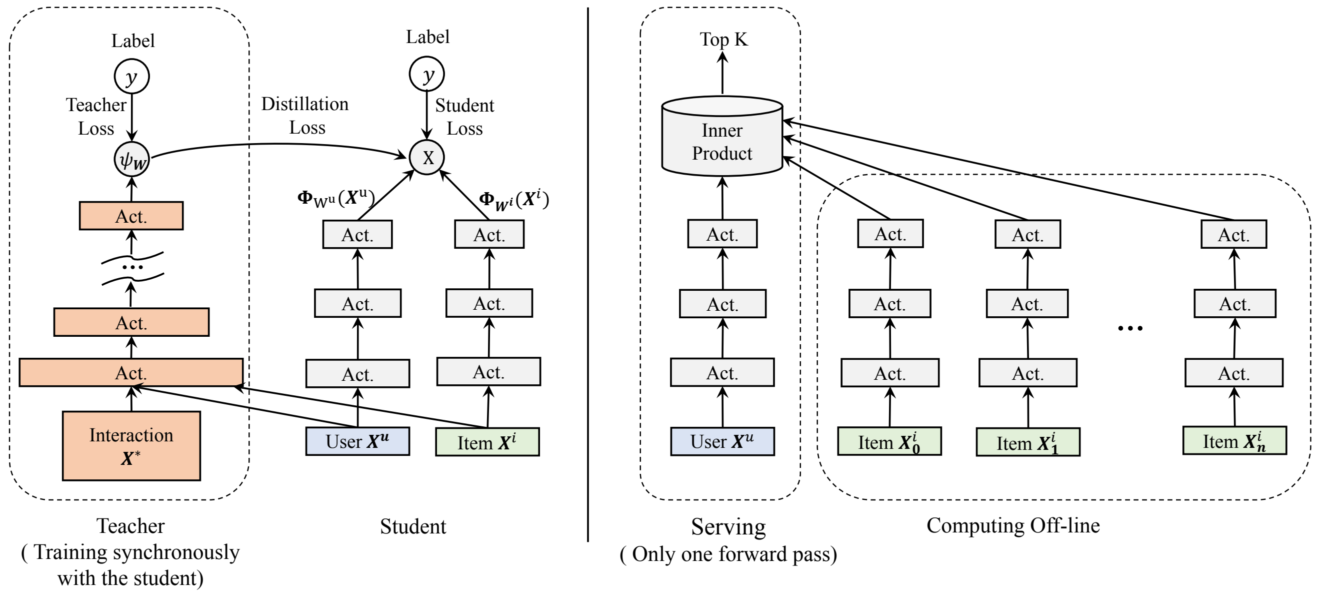
4 PRIVILEGED FEATURE DISTILLATION

Now we are to introduce our PFD. In the original privileged information distillation of Eq.(2), the teacher only processes the privilege information $\boldsymbol{X}^*$. This is well suited for action recognition [8] where X^* and X are in different modal. While in this work, we learn embeddings for all features. The privileged features and the regular ones can be simply combined to form a stronger teacher. In PFD, we thus modify the original function in Eq.(2) by adding regular inputs to the teacher, i.e.,

$$\min_{\boldsymbol{W}_s} L_s(\boldsymbol{y}, f(\boldsymbol{X}; \boldsymbol{W}_s)) + \lambda * L_d\left(f(\boldsymbol{X}, \boldsymbol{X}^*; \boldsymbol{W}_t), f(\boldsymbol{X}; \boldsymbol{W}_s)\right), \quad (4)$$

where the function of the teacher $f(\boldsymbol{X}, \boldsymbol{X}^*; \boldsymbol{W}_t)$ is trained in advance. In our applications, training the teacher model alone would take tens of days to converge. This is quite in-practical to apply distillation as Eq.(4). A more plausible way is to train the teacher and the student synchronously as in [1, 35, 36]. The objective function is then modified as follows:

$$\begin{aligned}\min_{\boldsymbol{W}_s, \boldsymbol{W}_t} \; & L_s(\boldsymbol{y}, f(\boldsymbol{X}; \boldsymbol{W}_s)) + \lambda * L_d\left(f(\boldsymbol{X}, \boldsymbol{X}^*; \boldsymbol{W}_t), f(\boldsymbol{X}; \boldsymbol{W}_s)\right) \\ & + L_t\left(\boldsymbol{y}, f(\boldsymbol{X}, \boldsymbol{X}^*; \boldsymbol{W}_t)\right)\end{aligned} \quad (5)$$



(a) Training with Unified Distillation (b) Serving with the Student

Figure 4: Illustration of training the inner-product model with unified distillation (UD) and (b) its deployment during serving. At the training time, the privileged features, i.e., interaction X^* between X^u and X^i, and the more complex DNN model together form a strong teacher to instruct the student. During serving, we compute the mappings $\Phi_{W^i}(\cdot)$ of all items off-line in advance. When a request comes, we only need to execute one forward pass to derive the user mapping $\Phi_{W^u}(X^u)$.

Although saving the training time, in our experiments, we find that synchronous training is un-stable, with chances to attain very bad results. This is mainly due to that the teacher is not well trained in the early stage. Its outputs could be noisy. By minimizing L_d in Eq.(5), the student might be distracted or even led to fail. To reduce side effect of such noisy teacher in early stage, a direct way is to decrease the value of λ. Here we adopt a warm update scheme to gradually increase λ with small initial λ_0 in the early stage. For better illustration, we summarize the method with adaptive λ in Algorithm 1. When computing the gradient with respect to the teacher parameters $\boldsymbol{W}_t$, we omit the distillation loss to avoid co-adaption between the teacher and student. Note that here we are using the stochastic gradient method only as an example. The adaptive scheme is well suited for all the state-of-art DNN optimizers.

Across this work, all models are trained in the parameter sever systems [6], where all parameters are stored in the servers and most computations are executed in the workers. The training speed is mainly depending on two aspects: the computation load of the workers and the communication volume between the workers and the servers. As indicated in Eq.(5), we are training the teacher and the student together. The learned parameters are roughly doubled. The communication volume between the servers and the workers is also doubled, which slows the training down. As the embeddings of all features take up most of the storage in the severs[3], here we propose to use shared embedding for the same feature between the teacher and student. As confirmed by the experiments below, adding such modification only slightly affects the performance while greatly speeds up the training. Besides, we only add a small portion of extra storage, i.e., the teacher network and the embeddings of privileged features, which makes the distillation technique can be easily incorporated into current systems.

Extension to Unified Distillation (UD). As illustrated in Figure 1, we are distilling the knowledge from the privileged features in PFD. While in MD, the knowledge is from the more complex teacher network. To further improve the distillation technique, a natural extension is to combine PFD with MD. Here we try to apply the unified distillation (UD) in the CTR estimation at coarse-grained ranking.

As Eq.(3) shows, we use the inner product model to increase the efficiency during serving. To some extent, the inner product model can be regarded as the generalized matrix factorization [4]. Although we are using *non-linear* mapping $\Phi_W(\cdot)$ to transform the user and item inputs, the model capacity is intrinsically limited by the *bi-linear* structure at the inner product operation. DNNs, with the capacity to approximate any function [5, 13], are considered as a substitution for the inner product model in the teacher. In fact, as proved in Theorem 1 of [20], the product operation can be approximated arbitrarily well by a two-layers neural network with only 4 neurons in the hidden layer. Thus the performance of using DNN is supposed to be lower-bounded by that of using the inner-product model.

In the CTR estimation at coarse-grained ranking, UD then adopts the DNN model as the teacher network. The inputs to the teacher, i.e., the privileged and regular features, are also preserved as PFD. In

[3]For the student model alone, the embeddings would take up to 100 Gigabytes of storage.

fact, the teacher here is the same as the structure for CTR estimation at fine-grained ranking. Thus UD in this task can be regarded as distilling knowledge from the fine-grained ranking to improve the coarse-grained ranking. For better illustration, we give the whole framework in Figure 4. During training, the inner product student is not only supervised by hard labels, but also by the soft labels produced from the DNN teacher. During serving, we extract the student part only, which relies on no privileged features. As the mappings $\Phi_{W^i}(X^i)$ of all items are independent of the users, we can compute them off-line in advance. When a request comes, the user mapping $\Phi_{W^u}(X^u)$ is firstly computed. After that, we compute its inner-product with the mappings of all items produced from the candidate generation stage. The top-k highest scored items are then chosen and fed to the fine-grained ranking. On the whole, we only execute one forward pass to derive the user mapping and conduct efficient inner product operations between the user and all candidates, which are rather friendly in the aspect of computational cost.

5 EXPERIMENTS

Now we are to conduct experiments to validate the effectiveness of distilling the privileged information. Here we adopt the Transformer [32] to model the user clicked/purchased history in Figure 1. We use one-layer Transformer with followed mean pooling layer in the time axis. The attained vector is then concatenated with all embeddings of the user features, which is regraded as the representation of the user. We also concatenate the embeddings of the item and the privileged features, as their corresponding representations. When feeding several types of inputs to the model, we simply concatenate their representations, too.

Across this work, we use LeakyReLU [23] as the activation for the DNN models and insert batch normalization [15] before the activation. The models are trained in the parameter servers with the asynchronous Adadelta optimizer [34]. In the first one million steps, the learning rate is increased linearly to the predefined value 0.01, which is then kept fixed across the updating. We set the batch size to 1024 and the number of epoch to 1. As introduced in Section 4, it is rather in-efficient to pre-train a teacher model. Here we adopt Algorithm 1 to train the teacher and student synchronously with adaptive λ. We initialize $\lambda_0 = 0$ and tune λ_{max} around the value 1.0 depending on tasks. At one million step, λ is increased to $\lambda_{max}/2$. At two million step, λ is increased to λ_{max} and kept fixed thereafter.

As the labels are in 1 or 0, i.e., whether the users clicked/purchased the item or not, we use the logloss for both the teacher and the student, i.e.,

$$L_{t/s} \triangleq \frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(1+e^{-f_{t/s,i}}\right) + (1-y_i)\log\left(1+e^{f_{t/s,i}}\right)\right), \quad (6)$$

where $f_{t/s,i}$ denotes the output of the i-th sample from the teacher or student model. For the distillation loss L_d, we use the cross entropy, i.e., by replacing y_i in the above equation with $1/(1+e^{-f_{t,i}})$. Here we measure the performance of models with the widely-used areas under the curve (AUC) in the next-day held-out data.

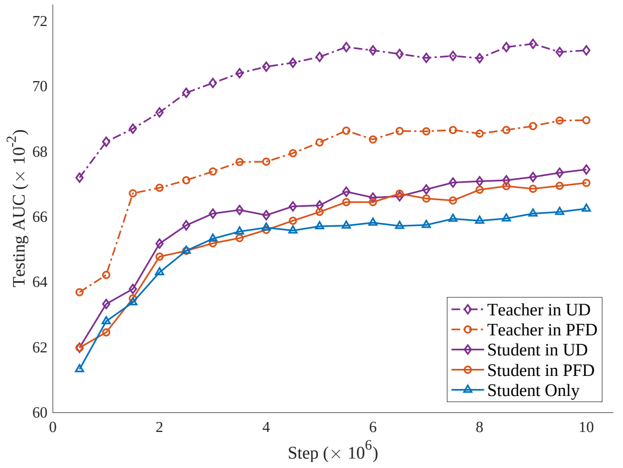


Figure 5: Testing AUC v.s. step of different models for CTR estimation at coarse-grained ranking. For better illustration, we do not add the curves from MD.

Table 1: Testing AUC ($\times 10^{-2}$) of different models for CTR estimation at coarse-grained ranking.

Models	$\sim 10^7$ Steps	$\sim 10^8$ Steps
Teacher in UD	71.1	74.1
Teacher in PFD	69.2	–
Teacher in MD	69.0	–
Student in UD	67.5	71.6
Student in PFD	67.1	–
Student in MD	67.0	–
Student Only	66.3	70.4

5.1 CTR at Coarse-grained Ranking

We first conduct experiments in the CTR estimation at coarse-grained ranking in Taobao recommendations. We use three layers of MLP as the user mapping $\Phi_{W^u}(\cdot)$ and the item mapping $\Phi_{W^i}(\cdot)$ in Eq.(3). The number of hidden neurons are set to 512, 256, and 128, respectively. In UD, we use four layers of MLP for the teacher model, with the number of hidden neurons being 1024, 512, 256, and 128, respectively. In PFD, we use the inner-product model for both the teacher and the student. And the interacted features are put at the user side of the teacher.

Overall performance. Here we test the performance of three distillation techniques, i.e., unified distillation (UD), privileged features distillation (PFD), and model distillation (MD). The testing AUC of different models are shown in the left column of Table 1. By comparing the teacher in PFD with the baseline without any distillation technique, we confirm the effectiveness of the interacted features. By distilling knowledge from these features, we improve the testing AUC of the student model, i.e., from 0.663 to 0.671. Note that in the industry, a steady 0.001 increase of AUC can be regarded as

Table 2: Testing AUC ($\times 10^{-2}$) of varying inner product dimensions. Overall, the improvement of UD preserves over different dimensions. Although dimension 192 achieves better performance, it increases the storage of item mappings as Figure 4 by 1.5×. Thus we still adopt 128 in our current systems.

Inner Product	Student in UD	Student Only
Dim. 64	67.2	66.1
Dim. 128	67.5	66.3
Dim. 192	67.8	66.7

significance given the huge number of clicks per day [33]. Empirically in our systems, +0.01 in the testing AUC will lead to around +4% in CTR. By combining PFD with MD, UD further improves the prediction performance of the student. The testing AUC is increased to 0.675. In order to validate that whether the advantage of UD over the original model could still hold when training longer with more data. We augment the training set in ×10. Due to the huge training cost, here we only execute UD and the baseline. As shown in the right part of Table 1, the student of UD still surpasses the original one with +0.012 AUC. We also plot the testing AUC v.s. step of different models in Figure 5. From the very beginning, the teachers are consistently achieving superior performance than the students. For student models using the distillation techniques, we can observe distinct gaps with the baseline, especially in the latter steps. In the first one million steps, the testing AUC of the three students is different, although we set the hyper-parameter $\lambda = 0$ as introduced earlier. This is mainly because that we use shared embeddings for the common features between the teacher and the student. Thus the student is also mildly affected.

Computational cost during inference by directly using interacted features. As discussed earlier, the interacted features are prohibited for the inner-product model during inference. Otherwise, we will need to execute the inference of the mappings $\Phi_{\boldsymbol{W}}(\cdot)$ as many times as the number of candidates, i.e., 10^5 here. In contrast, without such features during serving, we only need to execute the inference of the mapping once and compute its inner product with all candidates. Here we give a more detailed illustration on the computational gap between getting the mapping $\mathbf{\Phi}_{\boldsymbol{W}}(\cdot)$ and executing the inner product operation. Suppose that the input to the mapping $\mathbf{\Phi}_{\boldsymbol{W}}(\cdot)$ is in 1024 dimension. Theoretically, to get one such mapping here will need $1024 \times 512 + 512 \times 256 + 256 \times 128 \approx 6.9 \times 10^5$ fused multiply-add flops. In comparision, executing one inner product operation on the mappings in 128 dimension only needs 128 flops, which is $\sim 5400\times$ less. We also conduct simulated experiments in the personal computer. We repeat 10^5 times to simulate the mapping inference and the inner product operation, which totally costs 89.695s and 0.108s, respectively. Getting the mapping is about 830× slower than executing the inner product operation.

Effect of the inner product dimension. We also conduct experiments by varying the final dimension of the inner product model, which could largely affect the performance of the intrinsically bilinear model. We conduct experiments on dimension 64, 128, and 192. Results are shown in Table 2, where the larger dimension can yield better performance. Despite of this, UD still largely improves the student model. Although achieving better performance for dimension 192, it extra needs 50% more storage to save the item mappings as Figure 4. Considering the huge number of item corpus, in our current systems, we still use 128 dimensions for the inner product model.

Effect of sharing embedding. We further conduct experiments to test the effect of using shared embedding for common features between the teacher and the student. The training speeds of student only, UD with shared embedding, and UD with separated embedding are 320 steps/s, 280 steps/s, and 200 steps/s, respectively. By using shared embedding, we narrow the speed gap of UD with the original model. Although UD with separated embedding can get additional +0.001 AUC, it is still preferred to use shared embedding as it only needs around half of the storage during training in the parameter severs meanwhile gets a 1.4× faster training speed.

Table 3: Testing AUC ($\times 10^{-2}$) of different models for CVR estimation at fine-grained ranking.

Models	Student	Teacher
PFD	89.0	96.0
Student Only	88.5	–

5.2 CVR at Fine-grained Ranking

We further conduct experiments in the CVR estimation at fine-grained ranking in Taobao recommendations. For both the teacher and the student, we use three layers of MLP, with the number of hidden neurons being 512, 256, and 128, respectively. As directly increasing the number of layers or the number of neurons for the neural network has no statically significant improvement, we do not conduct MD and UD here.

Overall performance. The overall performance of using PFD is shown in Table 3. By utilizing PFD, we improve the baseline with +0.005 testing AUC. Empirically, in our systems, such improvement can lead to about +1.5% in CVR. In Figure 6, we also plot the curves of testing AUC v.s. step of different models. By utilizing PFD, the student consistently produces higher testing AUC than the baseline across the updating. After 2 million steps, the teacher model almost converges, which is mainly because that the post-event features, e.g., the dwell time on the detailed page, are highly predictive for CVR estimation.

Effect of sharing embedding. We also conduct experiments to test the effect of using shared embedding. The training speeds of student only, PFD with shared embedding, and PFD with separated embedding are $\sim$ 400 steps/s, $\sim$ 360 steps/s, and $\sim$ 280 steps/s, respectively. Besides training faster, PFD with shared embedding surpasses the counterpart with separated embedding by +0.001 testing AUC.

6 CONCLUSION

In this work, we target at the feature aspect in the prediction tasks of e-commerce recommendations. More specifically, we target at the

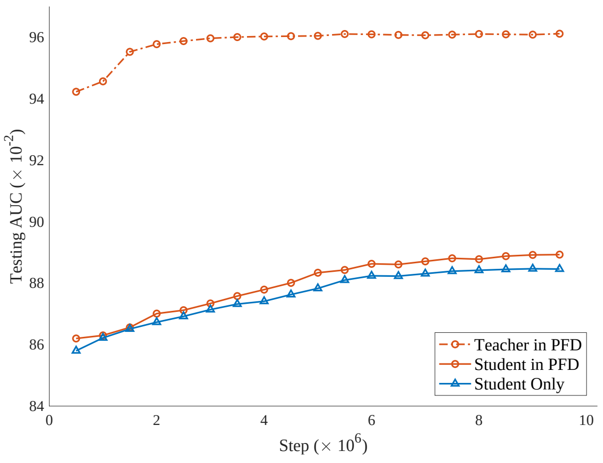


Figure 6: Testing AUC v.s. step of different models for CVR estimation at fine-grained ranking.

privileged features that are discriminative for the prediction while only available at the training time. As far as we know, such features are all neglected in current recommendation systems. By contrast, here we propose privileged features distillation (PFD) to make full use of them. During training, PFD helps the original model, i.e., the student, to learn better by transferring the knowledge distilled from the privileged features. While at serving, the student relies on no such features. PFD is complementary to the widely used model distillation. By combining both of the techniques we are able to achieve better performance further.

We conduct experiments on two fundamental prediction tasks in Taobao recommendations, i.e., CTR at coarse-grained ranking and CVR at fine-grained ranking. By distilling the interacted features that are prohibited(due to response time limit) for the inner product CTR model during serving and the post-event features that is only available after CVR estimation is done, respectively, PFD improves both of the strong baselines. After addressing several issues of training PFD, we can achieve comparable training speed as the baselines without any distillation.

REFERENCES

[1] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training through online distillation. In *ICLR*.
[2] Cristian BuciluǢ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 535–541.
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, NY, USA, 7–10.
[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. ACM, 191–198.
[5] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* 2, 4 (1989), 303–314.
[6] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. 2012. Large scale distributed deep networks. In *NIPS*. 1223–1231.
[7] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.
[8] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. 2018. Modality distillation with multiple stream networks for action recognition. In *ECCV*. 103–118.
[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *AAAI*. AAAI Press, 1725–1731.
[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.
[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).
[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[13] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4, 2 (1991), 251–257.
[14] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*. ACM, 2333–2338.
[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*. 448–456.
[16] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.
[17] Yoon Kim and Alexander M. Rush. 2016. Sequence-Level Knowledge Distillation. In *EMNLP*. 1317–1327.
[18] John Lambert, Ozan Sener, and Silvio Savarese. 2018. Deep learning under privileged information using heteroscedastic dropout. In *CVPR*. 8886–8895.
[19] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, NY, USA, 1754–1763.
[20] Henry W Lin, Max Tegmark, and David Rolnick. 2017. Why does deep and cheap learning work so well? *Journal of Statistical Physics* 168, 6 (2017), 1223–1247.
[21] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1557–1565.
[22] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2016. Unifying distillation and privileged information. In *ICLR*.
[23] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *ICML*, Vol. 30. 3.
[24] Asit Mishra and Debbie Marr. 2018. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *ICLR*.
[25] Yabo Ni, Dan Ou, Shichen Liu, Xiang Li, Wenwu Ou, Anxiang Zeng, and Luo Si. 2018. Perceive Your Users in Depth: Learning Universal User Representations from Multiple E-commerce Tasks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 596–605.
[26] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv:1701.06548* (2017).
[27] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. In *ICLR*.
[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *CVPR*. 2818–2826.
[29] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2289–2298.
[30] Vladimir Vapnik and Rauf Izmailov. 2015. Learning using privileged information: similarity control and knowledge transfer. *Journal of Machine Learning Research* 16, 2023-2049 (2015), 2.
[31] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural Networks* 22, 5-6 (2009), 544–557.
[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.
[33] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the International Workshop on Data Mining for Online Advertising*. ACM, NY, USA, 12.
[34] Matthew D Zeiler. 2012. ADADELTA: An adaptive learning rate method. *arXiv:1212.5701* (2012).
[35] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *CVPR*. 4320–4328.
[36] Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. 2018. Rocket launching: A universal and efficient framework for training well-performing light net. In *AAAI*.
[37] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1059–1068.