Ensembled CTR Prediction via Knowledge Distillation

Jieming Zhu
Huawei Noah's Ark Lab
Shenzhen, China
jmzhu@ieee.org

Jinyang Liu
School of Data and Computer Science
Sun Yat-Sen University, China
liujy57@mail2.sysu.edu.cn

Weiqi Li*
School of Data and Computer Science
Sun Yat-Sen University, China
liwq23@mail2.sysu.edu.cn

Jincai Lai
Huawei Noah's Ark Lab
Shenzhen, China
laijincai@huawei.com

Xiuqiang He
Huawei Noah's Ark Lab
Shenzhen, China
hexiuqiang1@huawei.com

Liang Chen, Zibin Zheng
School of Data and Computer Science
Sun Yat-Sen University, China
{chenliang6,zhzibin}@mail.sysu.edu.cn

ABSTRACT

Recently, deep learning-based models have been widely studied for click-through rate (CTR) prediction and lead to improved prediction accuracy in many industrial applications. However, current research focuses primarily on building complex network architectures to better capture sophisticated feature interactions and dynamic user behaviors. The increased model complexity may slow down online inference and hinder its adoption in real-time applications. Instead, our work targets at a new model training strategy based on knowledge distillation (KD). KD is a teacher-student learning framework to transfer knowledge learned from a teacher model to a student model. The KD strategy not only allows us to simplify the student model as a vanilla DNN model but also achieves significant accuracy improvements over the state-of-the-art teacher models. The benefits thus motivate us to further explore the use of a powerful ensemble of teachers for more accurate student model training. We also propose some novel techniques to facilitate ensembled CTR prediction, including teacher gating and early stopping by distillation loss. We conduct comprehensive experiments against 12 existing models and across three industrial datasets. Both offline and online A/B testing results show the effectiveness of our KD-based training strategy.

CCS CONCEPTS

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

KEYWORDS

CTR prediction, recommender systems, online advertising, knowledge distillation, model ensemble

*Work done during internship at Huawei.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412704

ACM Reference Format:
Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, and Liang Chen, Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412704

1 INTRODUCTION

In many applications such as recommender systems, online advertising, and Web search, click-through rate (CTR) is a key factor in business valuation, which measures the probability that users will click (or interact with) the promoted items. CTR prediction is extremely important because, for applications with a large user base, even a small improvement of prediction accuracy can potentially lead to a large increase of the overall revenue. For example, existing studies from Google [4, 26] and Microsoft [17] reveal that an absolute improvement of 1‰ in AUC or logloss is considered as practically significant in industrial-scale CTR prediction problems.

With the recent success of deep learning, many deep models have been proposed and gradually adopted in industry, such as Google's Wide&Deep [4], Huawei's DeepFM [8], Google's DCN [26], and Microsoft's xDeepFM [16]. They have demonstrated remarkable performance gains in practice. Current research for CTR prediction has two trends. First, deep models tend to become more and more complex in network architectures, for example, by employing convolution networks [18], recurrent networks [7], attention mechanisms [24, 32], and graph neural networks [15] to better capture sophisticated feature interactions and dynamic user behaviors. Second, following the popular wide&deep learning framework [4], many deep models employ a form of ensemble of two different sub-models to improve CTR prediction. Typical examples include DeepFM [8], DCN [26], xDeepFM [16], AutoInt+ [24], etc. Although these complex model architectures and ensembles lead to improved prediction accuracy, they may slow down model inference and even hinder the adoption in real-time applications. How to make use of a powerful model ensemble to achieve the best possible level of accuracy yet still retain the model complexity for online inference is the main subject of this work.

Different from the aforementioned research work that focuses primarily on model design, in this paper, we propose a model training strategy based on knowledge distillation (KD) [10]. KD is a teacher-student learning framework that guides the training of a student model with the knowledge distilled from a teacher model. Then, the student model is expected to achieve better accuracy

than it would if trained directly. While KD has been traditionally applied for model compression [10], we do not intentionally limit the student model size in terms of hidden layers and hidden units. We show that the training strategy not only allows to simplify the student model as a vanilla DNN model without sophisticated architectures, but also leads to significant accuracy improvements over the state-of-the-art teacher models (e.g., DeepFM [8], DCN [26], and xDeepFM [16]). Furthermore, the use of KD allows flexibility to develop teacher and student models of different architectures while only the student model is required for online serving.

These benefits motivate us to further explore the powerful ensemble of individual models as teachers, which potentially yields a more accurate student model. Our goal is to provide a KD-based training strategy that is generally applicable to many models for ensembled CTR prediction. Towards this goal, we have compared different KD schemes (soft label vs hint regression) and training schemes (pretrain vs co-train) for CTR prediction. While KD is an existing technique, we make the following extensions: 1) We propose a *teacher gating* network that enables sample-wise teacher selection to learn from multiple teachers adaptively. 2) We propose the novel use of distillation loss as the signal for *early stopping*, which not only alleviates overfitting but also enhances utilization of validation data during model training.

We evaluate our KD strategy comprehensively on three industrial-scale datasets: two open benchmarks (i.e., Criteo and Avazu), and a production dataset from Huawei's App store. The experimental results show that after training with KD, the student model not only attains better accuracy than itself but also can surpass the teacher model. Our ensemble distillation achieves the largest offline improvement reported so far (over 1% AUC improvement against DeepFM and DCN on Avazu). An online A/B test shows that our KD-based model achieves an average of 6.5% improvement in CTR and 8.2% improvement in eCPM (w.r.t revenue) in online traffic.

In summary, the main contributions of this work are as follows:

- Our work provides a comprehensive evaluation on the effectiveness of different KD schemes on the CTR prediction task, which serves as a guideline for potential industrial applications of KD.
- Our work makes the first attempt to apply KD for ensembled CTR prediction. We also propose novel extensions to improve the accuracy of the student model.
- We conduct both offline evaluation and online A/B testing. The results confirm the effectiveness of our KD-based training strategy.

In what follows, Section 2 introduces the background knowledge. Section 3 describes the details of our approach. The experimental results are reported in Section 4. We review the related work in Section 5 and finally conclude the paper in Section 6.

2 BACKGROUND

In this section, we briefly introduce the background of CTR prediction and knowledge distillation.

2.1 CTR Prediction

The objective of CTR prediction is to predict the probability a user will click a candidate item. Formally, we define it as $\hat{y} = \sigma(\phi(x))$, where $\phi(x)$ is the model function that outputs the logit (un-normalized log probability) value given input features x. $\sigma(\cdot)$ is the sigmoid function to map $\hat{y}$ to $[0, 1]$.

Deep models are powerful in capturing sophisticated high-order feature interactions to make accurate prediction. In the following, we choose four of the most representative deep models, i.e., Wide&Deep [4], DeepFM [8], DCN [26], and xDeepFM [16], since they have been successfully adopted in industry. As described in Equation 1~4, all these models follow the general wide and deep learning framework, which comprises a summation of two parts: one is a deep model $\phi_{DNN}(\mathbf{x})$, and the other is a wide model. Specifically, ϕ_{LR} and ϕ_{FM} are widely-used shallow models, while ϕ_{Cross} and ϕ_{CIN} are designed to capture bit-wise and vector-wise feature interactions, respectively.

$$Wide\&Deep: \ \phi_{WDL}(\mathbf{x}) = \phi_{LR}(\mathbf{x}) + \phi_{DNN}(\mathbf{x}) \tag{1}$$

$$DeepFM: \ \phi_{DeepFM}(\mathbf{x}) = \phi_{FM}(\mathbf{x}) + \phi_{DNN}(\mathbf{x}) \tag{2}$$

$$DCN: \ \phi_{DCN}(\mathbf{x}) = \phi_{Cross}(\mathbf{x}) + \phi_{DNN}(\mathbf{x}) \tag{3}$$

$$xDeepFM: \ \phi_{xDeepFM}(\mathbf{x}) = \phi_{CIN}(\mathbf{x}) + \phi_{DNN}(\mathbf{x}) \tag{4}$$

Finally, each model is trained to minimize the binary cross-entropy loss:

$$\mathcal{L}_{CE}(y, \hat{y}) = -\sum_i \left(y_i log\hat{y}_i + (1 - y_i)log(1 - \hat{y}_i)\right) . \tag{5}$$

We observe that current models tend to become more and more complex in order to capture effective high-order feature interactions. For example, xDeepFM runs about 2x slower than DeepFM with the use of outer products and convolutions. Although these models use a type of ensemble of two sub-models to boost prediction accuracy, directly ensembling more models is usually too complex to apply to real-time CTR prediction problems. Our work proposes a KD-based training strategy to enable the production use of the powerful ensemble models.

2.2 Knowledge Distillation

Knowledge distillation (KD) [10] is a type of teacher-student learning framework. The core idea of KD is to train a student model with supervision from not only the true labels but also the guidance provided by the teacher model (e.g., mimicking the output of a teacher model). The early goal of KD is for model compression [10, 19], where a compact model with fewer parameters is trained to match the accuracy of a large model. The success of KD has led to a variety of applications in image classification [19] and machine translation [25]. Our work makes the first attempt to apply KD for ensembled CTR prediction. Instead of compressing model size only, we aim to train a unified model from different teacher models or their ensembles for more accurate CTR prediction.

3 ENSEMBLED CTR PREDICTION

In this section, we describe the details of knowledge distillation for ensembled CTR prediction.

3.1 Overview

Figure 1 illustrates the framework of knowledge distillation, which consists of a teacher model and a student model. Let T be a teacher model that maps the input features x to the click probability $\hat{y}_T$.

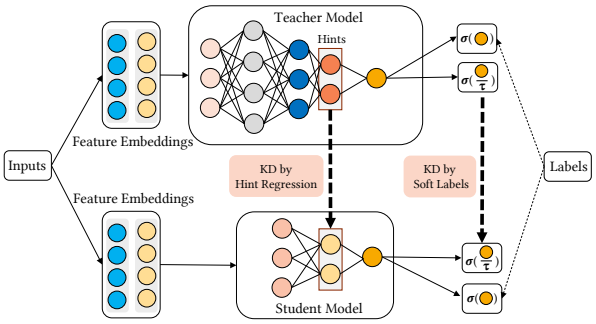


Figure 1: The framework of knowledge distillation.

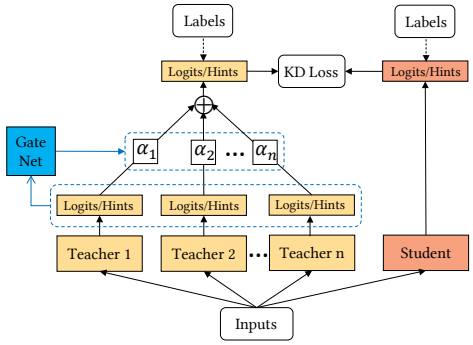


Figure 2: The framework of adaptive ensemble distillation

The same features are fed to the student model S and it predicts $\hat{y}_S$. During training, the loss functions for the teacher and the student models are formulated as follows:

$$\mathcal{L}_{\mathcal{T}} = \mathcal{L}_{CE}(y, \hat{y}_T) \tag{6}$$

$$\mathcal{L}_{\mathcal{S}} = \gamma \mathcal{L}_{CE}(y, \hat{y}_S) + \beta \mathcal{L}_{KD}(S, T) \tag{7}$$

where $\mathcal{L}_{CE}(\cdot)$ denotes the binary cross-entropy loss in Equation 5 and $\mathcal{L}_{KD}(S, T)$ denotes the distillation loss (defined later in Section 3.2). Especially for the student loss $\mathcal{L}_{\mathcal{S}}$, the first term $\mathcal{L}_{CE}$ captures the supervision from true labels and the second term $\mathcal{L}_{KD}$ measures the discrepancy between the student and the teacher models. β and γ are hyper-parameters to balance the weights of these two terms. Normally, we set $\beta + \gamma = 1$.

The KD framework allows flexibility to choose any teacher and student model architectures. More importantly, since only the student model is required for online inference, our KD-based training strategy does not disturb the process of model serving. In this work, we intentionally use a vanilla DNN network as the student model to demonstrate the effectiveness of KD. We intend to show that even a simple DNN model can learn well given useful guidance from a "good" teacher.

3.2 Distillation from One Teacher

To enable knowledge transfer from a teacher model to a student model, many different KD methods have been proposed. In this paper, we focus on the two most common methods: soft label [10] and hint regression [22].

3.2.1 Soft label. In this method, the student not only matches the ground-truth labels (i.e., hard labels), but also the probability outputs of the teacher model (i.e., soft labels). In contrast to hard labels, soft labels convey the subtle difference between two samples and therefore can help the student model generalize better than directly learning from hard labels.

As shown in Figure 1, the "KD by soft labels" method has the following distillation loss to penalize the discrepancy between the teacher and the student models.

$$\mathcal{L}_{KD}(S, T) = \mathcal{L}_{CE}\big(\sigma(\frac{z_T}{\tau}), \sigma(\frac{z_S}{\tau})\big) , \tag{8}$$

where $\mathcal{L}_{CE}$ is the binary cross-entropy loss defined in Equation 5. z_T and z_S denote the logits of both models. A temperature parameter τ is further applied to producing a softer probability distribution of labels. Finally, substituting the distillation loss $\mathcal{L}_{KD}(S, T)$ in Equation 7 deduces the final loss for the student model, which combines the supervision of hard labels and soft labels.

3.2.2 Hint regression. While soft labels provide direct guidance to the outputs of a student model, hint regression aims to guide the student's learning of representations. As "KD by hint regression" shown in Figure 1, a hint vector $\mathbf{v}_T$ is defined as an intermediate representation vector from a teacher's hidden layer. Likewise, we choose the student's representation vector $\mathbf{v}_S$ and force $\mathbf{v}_S$ to approximate $\mathbf{v}_T$ with linear regression.

$$\mathcal{L}_{KD}(S, T) = \|W\mathbf{v}_T - \mathbf{v}_S\|_2^2 , \tag{9}$$

where $W \in R^{n \times m}$ is a transformation matrix in case that $\mathbf{v}_T \in R^m$ and $\mathbf{v}_S \in R^n$ have different sizes. To some extent, the distillation loss $\mathcal{L}_{KD}$ acts as a form of regularization to the student model and helps the student mimic the learning process of the teacher. Accordingly, substituting the distillation loss $\mathcal{L}_{KD}(S, T)$ in Equation 7 derives the final loss for training the student model.

3.3 Distillation from Multiple Teachers

Model ensemble is a powerful technique to improve prediction accuracy and has become a dominant solution to win various recommendation competitions (e.g., Kaggle [2], Netflix [1]). However, directly applying model ensemble is often impractical due to the high model complexity for both training and inference. To make it possible, we extend our KD framework from a single teacher to multiple teachers.

One straightforward way is to average individual teacher models to make a stronger ensemble teacher, so that the problem reduces to learning from a single teacher. However, due to the different model architectures and training schemes used in practice, not all teachers can provide equally important knowledge on each sample. An ineffective teacher may even misguide the learning of the student. To gain effective knowledge from multiple teachers, we propose an adaptive ensemble distillation framework, as illustrated in Figure 2, to dynamically adjust their contributions. Formally, we have the following adaptive distillation loss:

$$\mathcal{L}_{KD}(S, T) = \mathcal{L}_{KD}(S, \sum_{i=1}^{M} \alpha_i T_i) , \tag{10}$$

where α_i is the importance weight for controlling the contribution of teacher T_i of M teachers. $\sum_{i=1}^{M} \alpha_i T_i$ denotes the ensemble of teachers, where the teacher weights should satisfy $\sum_{i=1}^{M} \alpha_i = 1$.

Teacher gating network. Instead of setting α_i as fixed parameters, we intend to learn α_i dynamically and make it adaptive to different data samples. To achieve this, we propose a teacher gating network to adjust the importance weights of teachers, which enables sample-wise teacher selection. More specifically, we employ a softmax function as the gating function.

$$\alpha_i = \frac{exp(w_i z_{T_i} + b_i)}{\sum_{i=1}^{M} exp(w_i z_{T_i} + b_i)}, \ i = 1, \cdots, M \tag{11}$$

where $\{w_i, b_i\}_{i=1}^{M}$ are the parameters to learn. In intuition, the gating network uses all teachers' outputs to determine the relative importance of each other.

3.4 Training

3.4.1 Training scheme. In this work, we investigate both the pre-train scheme and the co-train scheme for KD. The pre-train scheme is to train the teacher and student models in two phases. For the co-train scheme, both teacher and student models are trained jointly while the back-propagation of the distillation loss is unidirectional so that the student model learns from the teacher, but not vice versa. The co-train scheme is faster than the two-phase pre-train scheme, but requires much more GPU memory. We also empirically compare the performance between pre-train and co-train schemes in Section 4.2. For the case of ensemble distillation, we focus mainly on the pre-train scheme, since the co-training scheme requires to load multiple teacher models into the GPU memory. We consider three state-of-the-art models (i.e., DeepFM, DCN, xDeepFM) as teachers and train each of them by minimizing the loss in Equation 6. When training the student model according to Equation 7, the teacher can provide better guidance to the student model. The teacher gating network is trained together with the student model for sample-wise teacher selection.

3.4.2 Early stopping via distillation loss. It is a common practice to employ early stopping to reduce overfitting. When training a teacher model, we employ the left-out validation set to monitor the metrics (e.g., AUC) for early stopping. When training a student model, we instead propose to use the distillation loss from the teacher model as the signal for early stopping. It works because the teacher model has been trained with reduced overfitting by early stopping on the validation set, and the student model can inherit this ability when mimicking the teacher model. This eliminates the need to retain the validation set and helps the student model generalize better by using the validation set (often the most recent data) to enrich the training data.

4 EXPERIMENTS

In this section, we report on our experimental results to evaluate the effectiveness of our KD-based training strategy for CTR prediction.

4.1 Experiment Setup

4.1.1 Datasets. We use three real-world datasets in our experiments: Criteo[1], Avazu[2], and our production dataset. Table 1 summarizes the data statistics information.

[1]https://www.kaggle.com/c/criteo-display-ad-challenge
[2]https://www.kaggle.com/c/avazu-ctr-prediction

Table 1: Dataset statistics

Dataset	#Instances	#Fields	#Features	Positive Ratio
Criteo	46M	39	1M	26%
Avazu	40M	22	972K	17%
Production	231M	29	160K	–

Criteo. The dataset consists of ad click logs over a week. It comprises 26 categorical feature fields and 13 numerical feature fields. Following Google's DCN work [26], we randomly split the data of the last two days into a validation set and a test set of equal size, while the rest is used for training.

Avazu. The dataset contains 10 days of click logs. It has a total of 22 fields with categorical features such as app id, app category, device id, etc. Following the recent AutoInt work [24], we randomly split the data into 8:1:1 as the training, validation, and test sets, respectively.

Production. We collected the production dataset from users' click logs over 8 days. It has 29 categorical feature fields, such as app id, category, tags, city, recent clicks, etc. Following our practice in production, we split the data sequentially, where the first 7 days are used for training and the last day is equally split for validation and testing, respectively.

For the two open datasets, we preprocess the data following the work in [24]. First, we replace the features less than a threshold with a default "<UNK>" ID, where the threshold is set to 10 and 5 for Criteo and Avazu, respectively. Second, numerical values in Criteo are normalized to avoid large variance by transforming each value x to $\log^2(x)$, if $x > 2$.

4.1.2 Baseline models. To compare the performance, we select a total of 12 representative models, including LR, FM [21], FFM [12], DNN [5], Wide&Deep [4], DeepFM [8], DCN [26], xDeepFM [8], PIN [20], FiBiNet [11], AutoInt+ [24], and FGCNN [18]. Although we cannot enumerate all the existing models, we have made a relatively comprehensive comparison in contrast to previous studies. To test statistical significance, in Section 4.5, we repeat the experiments 5 times by changing random seeds, and run the two-tailed paired t-test.

4.1.3 Implementation details. All the models are implemented in PyTorch. We use Adam for optimization, and set the batch size to 2000. The learning rate is set to 0.001. Categorical features are embedded to dimensions of 20, 40, and 40 for Criteo, Avazu, and Production, respectively. We use grid search to tune other hype-parameters. Specifically, we apply L_2 regularization on feature embeddings and search the regularization weight in $[0, 1e-8, 1e-7, 1e-6, 1e-5]$. We also apply dropout to hidden layers of DNN with dropout rates selected in $[0.1, 0.2, 0.3, 0.4, 0.5]$. The number of hidden layers is set among $[2, 3, 4, 5, 6]$. We keep the same size of each layer and select it from $[300, 400, 500, 600]$. Especially, we vary the layers of the cross net (in DCN) and CIN (in xDeepFM) among $[1, 2, 3, 4, 5]$. As for KD by soft labels, we set $\beta + \gamma = 1$ and tune β from 0 to 1 with a step of 0.1. For KD by hint regression, we set $\gamma = 1$ and tune β in the range $[0, 1e-6, 1e-5, 1e-4, 1e-3]$. We also try different values of temperature τ among $[1, 2, 3, 4, 5, 7, 10, 15]$. To avoid overfitting, early stopping is adopted when the metrics on the validation set (for teacher training) or the distillation loss (for student training) stop improving in three consecutive epochs.

Table 2: Performance of different KD schemes. T→S indicates that T is a teacher and S is a student.

Model	KD Scheme	Avazu		Criteo	
		AUC	Logloss	AUC	Logloss
DNN		0.7740	0.3825	0.8006	0.4551
DeepFM→DNN	hint+co-train	0.7791	0.3805	0.8028	0.4525
	hint+pre-train	0.7769	0.3822	0.8030	0.4527
	soft label+co-train	0.7773	0.3807	0.8031	0.4524
	soft label+pre-train	**0.7802**	**0.3800**	**0.8039**	**0.4517**
DCN→DNN	hint+co-train	0.7778	0.3808	0.8032	0.4521
	hint+pre-train	0.7784	0.3805	0.8031	0.4521
	soft label+co-train	0.7782	0.3812	0.8035	0.4518
	soft label+pre-train	**0.7794**	**0.3800**	**0.8040**	**0.4518**
xDeepFM→DNN	hint+co-train	0.7795	0.3802	0.8032	0.4525
	hint+pre-train	0.7775	0.3809	0.8021	0.4531
	soft label+co-train	0.7762	0.3815	0.8034	0.4525
	soft label+pre-train	**0.7797**	**0.3797**	**0.8039**	**0.4517**

4.2 Performance of Different KD Schemes

In this section, we aim to study the research question of which KD scheme performs the best for CTR prediction. Here, we define a candidate KD scheme as a combination of the KD methods (soft label v.s. hint regression) and training methods (pre-train v.s. co-train). To study the effect of KD schemes only, in this experiment, we fix the size of the student DNN model as 500×5. Each model is tuned to attain its best performance for Avazu and Criteo, respectively.

Table 2 shows the results of different KD schemes. Especially, the first row indicates the student-only performance, i.e., training the DNN directly. We can make the following observations from the results: 1) All student models trained via different KD schemes consistently outperform the student-only DNN by a large margin. 2) The KD scheme of "soft label + pre-train" performs the best among all candidate KD schemes. However, the difference between these schemes is small, making it a flexible choice to select an appropriate KD scheme. In practice, the co-train scheme requires to load both of the teacher model and the student model into the GPU memory, which hinders its scalability especially when multiple teacher models are available. Therefore, we focus mainly on the "soft label + pre-train" scheme in the following experiments.

4.3 Performance across Different Teachers and Students

In this experiment, we intend to evaluate what performance could be achieved by applying different teacher models and student models. In particular, we select five teacher models including DNN, DeepFM, DCN, xDeepFM, and 3T (an ensemble of DeepFM + DCN + xDeepFM). We first tune each teacher model to attain its best performance, and report the teachers' results as the row "w/o KD" in Table 3. The results of 3T are an average of the three models. Then, we explore different student models (i.e., DNN, DeepFM, DCN, xDeepFM) and perform KD experiments in a total of 20 settings (5 teachers and 4 students). We fix the teacher models and tune the student models.

The experimental results are reported in row 2~5 in Table 3. We observe that: 1) All student models trained with KD achieve even better performance than those using teachers only. The last row shows the maximal improvements (absolute value in ‰) over the teacher's performance. This finding (i.e., students beating teachers) is suprising since it is rarely reported in traditional KD studies, where a student model is usually cut to a small size for model compression. We further evaluate the impact of student model size on performance in Section 4.7. 2) With our KD strategy, even the vanilla DNN model can learn well, surpassing the state-of-the-art complex teacher models. This confirms the good model capacity of DNN and the effectiveness of KD to guide the learning. Using DeepFM or DCN as a student can even attain better performance. This shows that better student architectures can be further explored to maximize the benefit of KD. But we focus mainly on the DNN model to demonstrate the superiority of KD, and leave the design of an optimal student model architecture for future research. 3) Compared to the cases of KD from one teacher, the performance is largely improved by using the 3T ensemble as teacher models. In total, 3T→DNN makes up to 12.5‰ and 5‰ absolute improvements in AUC over a single teacher model for Avazu and Criteo, respectively. The encouraging results motivate us to explore more towards ensemble distillation.

4.4 Performance of Different Teacher Ensembles

After confirming the superiority of using 3T as an ensemble of teachers, we intend to evaluate the detailed performance of using different numbers and different types of teacher ensembles. As shown in Table 4, we experiment with different number of teachers from 1T to 6T, and report both the results of teacher ensembles and their students. For the case 1T, the results correspond to using DCN and xDeepFM as the single teacher for Avazu and Criteo respectively, due to their best performance. For the cases of multiple teachers (3T and 6T), we study two types of training methods (denoted as M and D) to generate teachers. Especially, 3T(M) denotes the combination "DeepFM + DCN + xDeepFM" with different model architectures. 6T(M) doubles 3T(M) with different random initialization seeds. 3T(D) and 6T(D) denote the combination of 3 or 6 models (DCN for Avazu and xDeepFM for Criteo) trained on different data partitions of training and validation sets, while the testing set remains the same. Again, we observe that student models trained with different teacher ensembles consistently outperform their teachers. Meanwhile, applying more teachers (1T→3T→6T) for ensemble distillation results in higher performance of student models. But the increase diminishes. In addition, teacher ensembles trained on different data partitions perform better (i.e., D better than M), which in turn leads to better students. This is partly due to the reason that we use the best-performing single teachers for 3T(D) and 6T(D) settings.

4.5 Comparison with the State-of-the-art Models

We make a comprehensive performance comparison between our DNN models trained with KD and the state-of-the-art models (12

Table 3: Performance across different teacher and student models. Each column represents a teacher model (T) and each row represents a student model (S). 3T denotes an ensemble of teachers: DeepFM + DCN + xDeepFM. The row "w/o KD" represents the performance of using teachers only. LL is short for logloss. The best results in each column are marked in bold.

	Avazu										Criteo									
T / S	DNN		DeepFM		DCN		xDeepFM		3T		DNN		DeepFM		DCN		xDeepFM		3T	
	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL	AUC	LL
w/o KD	0.7740	0.3825	0.7763	0.3814	0.7774	0.3811	0.7770	0.3811	0.7804	0.3788	0.8006	0.4551	0.8031	0.4522	0.8030	0.4524	0.8036	0.4518	0.8046	0.4508
DNN	0.7825	0.3780	0.7818	0.3786	0.7825	0.3779	0.7820	0.3782	0.7865	0.3757	**0.8038**	**0.4517**	0.8040	0.4516	0.8038	0.4516	0.8042	0.4513	0.8056	0.4500
DeepFM	0.7825	0.3781	0.7837	0.3777	0.7834	0.3778	0.7837	0.3779	0.7860	0.3758	0.8037	0.4518	**0.8041**	**0.4516**	**0.8041**	**0.4514**	0.8042	0.4514	0.8057	0.4499
DCN	**0.7848**	**0.3768**	**0.7850**	**0.3768**	**0.7854**	**0.3771**	**0.7848**	**0.3771**	**0.7871**	**0.3753**	0.8037	0.4528	0.8038	0.4516	0.8041	0.4516	**0.8043**	**0.4513**	**0.8057**	**0.4498**
xDeepFM	0.7825	0.3780	0.7831	0.3777	0.7829	0.3777	0.7823	0.3785	0.7853	0.3761	0.8035	0.4521	0.8034	0.4519	0.8035	0.4523	0.8039	0.4516	0.8057	0.4499
Max Impr.	10.8‰	5.7‰	8.7‰	4.6‰	8.0‰	4.0‰	7.8‰	4.0‰	6.7‰	3.3‰	3.2‰	3.4‰	1.0‰	0.6‰	1.1‰	1.0‰	0.7‰	0.5‰	1.1‰	1.0‰

Table 4: Performance of different teacher ensembles. xT indicates x teachers. M and D denote two types of ensemble model training, via different model architectures (M) or different data partitions (D). The best results in each setting is marked in bold.

#Teachers	Model	Avazu		Criteo	
		AUC	Logloss	AUC	Logloss
1T	1T	0.7774	0.3811	0.8036	0.4518
	1T→DNN	0.7825	0.3779	0.8042	0.4513
3T	3T(M)	0.7804	0.3788	0.8046	0.4508
	3T(M)→DNN	0.7865	0.3757	0.8056	0.4500
	3T(D)	0.7871	0.3754	0.8071	0.4488
	3T(D)→DNN	**0.7879**	**0.3751**	**0.8073**	**0.4485**
6T	6T(M)	0.7816	0.3784	0.8059	0.4497
	6T(M)→DNN	0.7872	0.3751	0.8061	0.4495
	6T(D)	0.7881	0.3746	0.8076	0.4481
	6T(D)→DNN	**0.7885**	**0.3745**	**0.8077**	**0.4480**

models in total). The upper part of Table 5 shows the overall performance of all the compared models across three datasets. In particular, the underlined numbers show the best baseline models and bold numbers are the best results among all the models. 1T and 3T in the table denote the best teacher models mentioned in Table 4. With a focus on evaluating KD on representative models, we did not reproduce all of the baseline models. Instead, we compare some recent deep models directly using the results reported in their papers, which are shown in the lower part of Table 5. For fairness, we only show the performance improvements over DeepFM.

We summarise our observations from the table as follows: 1) Deep models generally outperform shallow models (LR, FM, and FFM), which reveals the capability of deep models to more capture complex feature interactions. 2) Models trained with KD (1T and 3T) consistently beat all baseline models with a large margin on all the three datasets. Specifically, 3T makes 10.5‰ , 3.7‰ , and 3.2‰ absolute improvements in AUC over the best-performing baseline models for the three datasets. 3) 3T makes further improvements compared to 1T on all datasets. This confirms the value of ensemble distillation, which makes the student models generalize better given the diversity of different teachers. 4) Compared to some more recently proposed deep models as shown in the lower part, the largest improvements from 3T also confirms the effectiveness of KD, considering that our student model is a vanilla DNN. Finally, we emphasize that a student model trained with KD can

Table 5: Performance comparison between our DNN models and the state-of-the-art models (The upper part shows our experimental results and the lower part reports the results from existing work).

The best baseline is underlined while our best result is marked in bold. * indicates $p < 0.001$ by two tailed paired t-test with 5 runs.

Model	Avazu		Criteo		Production	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
LR	0.7661	0.3878	0.7843	0.4689	0.9012	0.1371
FM	0.7754	0.3827	0.7965	0.4586	0.9326	0.1227
FFM	0.7765	0.3814	0.7997	0.4558	0.9358	0.1170
DNN	0.7740	0.3825	0.8006	0.4551	0.9381	0.1153
Wide&Deep	0.7760	0.3816	0.8028	0.4524	0.9389	0.1146
DeepFM	0.7763	0.3814	0.8031	0.4522	0.9391	0.1145
DCN	0.7774	0.3811	0.8030	0.4524	0.9392	0.1147
xDeepFM	0.7770	0.3811	0.8036	0.4518	0.9391	0.1146
1T→DNN	0.7825	0.3779	0.8042	0.4513	0.9411	0.1131
3T→DNN	**0.7879**	**0.3751**	**0.8073**	**0.4485**	**0.9424**	**0.1119**
Max Impr.	10.5‰*	6.0‰*	3.7‰*	3.3‰*	3.2‰*	2.6‰*

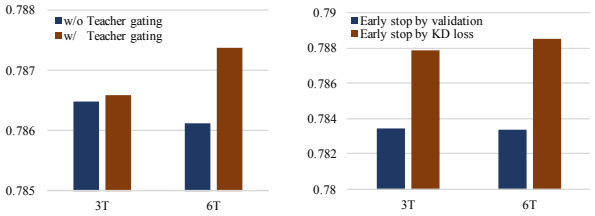
A comparison of some recent deep models with respect to the performance improvements over DeepFM (absolute values in ‰).

Model	Avazu		Criteo	
	AUC	Logloss	AUC	Logloss
PIN	3.6‰	2.2‰	3.0‰	3.3‰
FiBiNet	4.6‰	2.4‰	1.8‰	2.2‰
AutoInt+	2.3‰	1.8‰	1.7‰	1.5‰
FGCNN	4.7‰	3.1‰	3.1‰	3.5‰
1T→DNN	6.0‰	2.8‰	1.1‰	0.9‰
3T→DNN	**11.5‰**	**6.3‰**	**4.2‰**	**3.5‰**

even surpass the ensembled teacher models is a surprising finding. It deserves our more investigation on the KD mechanism to help reduce overfitting in future.

4.6 Ablation Study

4.6.1 Teacher gating. We evaluate the effectiveness of teacher gating by comparing the performance of training with gating or without gating (i.e., via averaging), in the case of 3T and 6T on Avazu. To demonstrate the capability of gating to distill effective knowledge from diverse teachers, in this experiment, we set 6T as an ensemble of DeepFM, xDeepFM, DCN, DNN, Wide&Deep and AutoInt+, and 3T as DeepFM, xDeepFM, and DCN. From the results shown in Figure 3(a), we see that the student model learns better with teacher



(a) Teacher gating (b) Early stop via distillation loss

Figure 3: Ablation study results in AUC

Table 6: Performance comparison of different student sizes. The upper part describes the model sizes and the lower part shows the corresponding results.

Model	Avazu	Criteo
DNN	300x3	400x3
DeepFM	DNN: 300x3	DNN: 500x4
DCN	DNN: 500x5, Cross: 2 layers	DNN: 500x4, Cross: 2 layers
xDeepFM	DNN: 500x5, CIN: 2 layers	DNN: 500x4, CIN: 1 layer
DeepFM→DNN+	DNN+: 500x5	DNN+: 500x6
DCN→DNN+	DNN+: 500x3	DNN+: 400x4
xDeepFM→DNN+	DNN+: 500x5	DNN+: 500x5

Model	Avazu		Criteo	
	AUC	Logloss	AUC	Logloss
DNN	0.7740	0.3825	0.8006	0.4551
DeepFM	0.7763	0.3814	0.8031	0.4522
DeepFM→DNN	0.7757	0.3819	0.8036	0.4518
DeepFM→DNN+	**0.7818**	**0.3786**	**0.8040**	**0.4516**
DCN	0.7774	0.3811	0.8030	0.4524
DCN→DNN	0.7824	0.3786	0.8035	0.4519
DCN→DNN+	**0.7825**	**0.3779**	**0.8038**	**0.4516**
xDeepFM	0.7770	0.3811	0.8036	0.4518
xDeepFM→DNN	0.7795	0.3800	0.8041	0.4517
xDeepFM→DNN+	**0.7820**	**0.3782**	**0.8042**	**0.4513**

gating, which indicates the usefulness of sample-wise teacher selection. It is worth noting that, without gating, 6T performs worse than 3T because 6T contains more diverse models. Teacher gating helps adjust the importance weights of teachers adaptively and thus makes 6T largely improved.

4.6.2 Early stopping via distillation loss. We compare the two ways of early stopping (using validation set v.s. using KD loss). The results in Figure 3(b) show that the KD loss between the teacher and student models serves a good monitor signal for early stopping. It allows full utilization of the validation data (usually more recent) for training the student model and thus obtains better performance.

4.7 Impact of student model size

The size of a DNN model, w.r.t. both hidden layers and hidden units, usually determines its model capacity. To study the impact of student model size, we first tune teacher models and keep them fixed (the same settings with Section 4.3). We then perform two group of experiments. In the first group, we set each student model to its best DNN size tuned in the last step (denoted as DNN). In the second group, we re-tune the size of each student DNN model to a larger size (denoted as DNN+). Table 6 presents the detailed model sizes in the upper part and the the corresponding results in the lower part. For instance, the DNN model alone attains the best performance at the size of 300x3 (3 hidden layers with 300 units each) on Avazu. However, when using DeepFM as a teacher for KD training, the 500x5 DNN+ performs better than the above 300x3 DNN. We can observe similar results in other settings from the table. The results imply that our goal is not directly model compression, since a larger student model tends to achieve better performance.

4.8 A/B Testing

We conducted an online A/B test in the app recommender system of Huawei's App store. It has hundreds of millions of daily active users which generates hundreds of billions of user feedback events everyday, such as browsing, clicking and downloading apps. In the online serving system, hundreds of candidate apps are first selected during the matching phrase, and then ranked by a ranking model (e.g., DCN) to generate the final recommendation list. In the A/B test, the baseline model in production is an extended version of DCN, demonstrating superiority over other competing models. As the model is re-trained on a daily basis, we choose the recent two old versions of models as ensemble teachers for efficiency consideration and then distill a student model of the same architecture on the newest data over recent 15 days for online serving. The A/B test was performed over one week, and we randomly select 5% of online traffic for both the control group and our experiment group. On average, our model improves the overall DLR (app downloading rate defined as *#download/#impressions*) by 6.5% and eCPM (expected cost per mille) by 8.2% over the baseline. This is a substantial improvement and demonstrates the effectiveness of our KD approach.

5 RELATED WORK

5.1 CTR Prediction

Since the successful application of DNNs to Youtube recommendation [5], deep learning has been widely studied for CTR prediction. Some work investigates the integration of DNNs and traditional shallow models (e.g., Wide&Deep [4], DeepFM [8]). Some models aim to capture different orders of feature interactions explicitly (e.g., DCN [26], xDeepFM [16]). Some models explore the use of convolutional networks (e.g., FGCNN [18]), recurrent networks (e.g., DSIN [7]), attention networks (e.g., AutoInt [24], FiBiNET [11]) and graph neural networks (e.g., FiGNN [15]) to learn high-order feature interactions. Some other studies focus on modeling evolutionary user interests in CTR prediction (e.g., DIN [32]). Our work demonstrates the effectiveness of KD using multiple representative models, but the framework is generally applicable to other models since they can be used as teacher models as well.

Although model ensemble is a powerful technique to boost prediction accuracy, the high complexity in such ensembles (e.g., over 100 models ensembled in the Netflix prize [1]) hinders the adoption in the industry. To restrict the model complexity for online deployment, most industrial studies reported by Facebook [9], Google [14], and Microsoft [13, 17] focus on the ensemble of only two models. Instead, our work tackles this issue via ensemble distillation and delivers a simplified yet equally powerful student model for online inference.

5.2 Knowledge Distillation

The knowledge distilled from a teacher varies in different forms. In addition to soft label [10] and hint regression [22] we used, some work further explores the knowledge transfer through inter-layer flow [29], cross-sample similarity [27], attention mechanism [30], etc. Some more recent efforts have been made towards ensemble distillation, facilitating a variety of tasks such as image classification [23] and machine translation [25]. For recommender systems, Chen et al. [3] propose an adversarial distillation approach to learn from external knowledge base for collaborative filtering. Xu et al. [28] proposes a KD approach to transfer privileged features from ranking to matching tasks. Liu et al. [6] leverage KD for debiasing in recommendation via uniform data. Zhang et al. [31] study the mutual learning between path-based and embedding based recommendation models. Our work is partially inspired from these studies and serves as the first attempt to apply KD for ensembled CTR prediction.

6 CONCLUSION

Model ensemble is a powerful technique to improve prediction accuracy. In this paper, we make an attempt to apply KD for ensembled CTR prediction. Our KD-based training strategy enables the production use of a powerful ensemble of teacher models and makes large accuracy improvements. Surprisingly, we demonstrate that a vanilla DNN trained with KD can even surpasses the ensemble teacher models. Our key contribution includes an intensive evaluation of the KD-based training strategy, teacher gating for sample-wise teacher selection, and early stopping by KD loss that increases the utilization of validation data. Both offline and online A/B testing results demonstrate the effectiveness of our approach. We hope that our encouraging results could attract more research efforts to study training strategies for CTR prediction.

7 ACKNOWLEDGEMENTS

The work described in this paper was partially supported by the Key-Area Research and Development Program of Guangdong Province (2018B010109001), and the National Natural Science Foundation of China (U1811462). Zibin Zheng is the corresponding author.

REFERENCES

[1] 2009. The BigChaos Solution to the Netflix Grand Prize. https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

[2] 2015. 4 Idiots' Approach for Click-through Rate Prediction. https://www.csie.ntu.edu.tw/~r01922136/slides/kaggle-avazu.pdf

[3] Xu Chen, Yongfeng Zhang, Hongteng Xu, Zheng Qin, and Hongyuan Zha. 2019. Adversarial Distillation for Efficient Recommendation with External Knowledge. *ACM Trans. Inf. Syst.* 37, 1 (2019), 12:1–12:28.

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys)*. 7–10.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 191–198.

[6] Zhenhua Dong Xiuqiang He Weike Pan Zhong Ming Dugang Liu, Pengxiang Cheng. 2020. A General Knowledge Distillation Framework for Counterfactual Recommendation via Uniform Data. In *International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR)*.

[7] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 2301–2307.

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *International Joint Conference on Artificial Intelligence (IJCAI)*. 1725–1731.

[9] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, et al. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising, (ADKDD)*. 5:1–5:9.

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[11] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys)*.

[12] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 43–50.

[13] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. 2019. DeepGBM: A Deep Learning Framework Distilled by GBDT for Online Prediction Tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 384–394.

[14] Pan Li, Zhen Qin, Xuanhui Wang, and Donald Metzler. 2019. Combining Decision Trees and Neural Networks for Learning-to-Rank in Personal Search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 2032–2040.

[15] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Fi-GNN: Modeling Feature Interactions via Graph Neural Networks for CTR Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 539–548.

[16] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, et al. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *International Conference on Knowledge Discovery & Data Mining (KDD)*. 1754–1763.

[17] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model Ensemble for Click Prediction in Bing Search Ads. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW)*.

[18] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference (WWW)*. ACM.

[19] Asit K. Mishra and Debbie Marr. 2018. Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. In *Proceedings of the 6th International Conference on Learning Representations, (ICLR)*.

[20] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *TOIS* (2019).

[21] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*. 995–1000.

[22] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. In *Proceedings of International Conference on Learning Representations, (ICLR)*.

[23] Zhiqiang Shen, Zhankui He, and Xiangyang Xue. 2019. MEAL: Multi-Model Ensemble via Adversarial Learning. In *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*. 4886–4893.

[24] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM, 1161–1170.

[25] Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. 2019. Multilingual Neural Machine Translation with Knowledge Distillation. In *7th International Conference on Learning Representations (ICLR)*.

[26] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the 11th International Workshop on Data Mining for Online Advertising (ADKDD)*. 12:1–12:7.

[27] Ancong Wu, Wei-Shi Zheng, Xiaowei Guo, and Jian-Huang Lai. 2019. Distilled Person Re-Identification: Towards a More Scalable System. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[28] Chen Xu, Quan Li, Junfeng Ge, Jinyang Gao, Xiaoyong Yang, Changhua Pei, Hanxiao Sun, and Wenwu Ou. 2019. Privileged Features Distillation for E-Commerce Recommendations. *CoRR* abs/1907.05171 (2019).

[29] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. 2017. A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[30] Sergey Zagoruyko and Nikos Komodakis. 2017. Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer. In *International Conference on Learning Representations, (ICLR) 2017*.

[31] Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling Structured Knowledge into Embeddings for Explainable and Accurate Recommendation. In *Conference on Web Search and Data Mining (WSDM)*. 735–743.

[32] Guorui Zhou, Xiaoqiang Zhu, Chengru 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 ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1059–1068.