ReLoop: A Self-Correction Continual Learning Loop for Recommender Systems

Guohao Cai[†]
Huawei Noah's Ark Lab
Shenzhen, China
caiguohao1@huawei.com

Jieming Zhu[†]
Huawei Noah's Ark Lab
Shenzhen, China
jamie.zhu@huawei.com

Quanyu Dai
Huawei Noah's Ark Lab
Shenzhen, China
daiquanyu@huawei.com

Zhenhua Dong[*]
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

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

Ruiming Tang
Huawei Noah's Ark Lab
Shenzhen, China
tangruiming@huawei.com

Rui Zhang[*]
www.ruizhang.info
rayteam@yeah.net

ABSTRACT

Deep learning-based recommendation has become a widely adopted technique in various online applications. Typically, a deployed model undergoes frequent re-training to capture users' dynamic behaviors from newly collected interaction logs. However, the current model training process only acquires users' feedbacks as labels, but fails to take into account the errors made in previous recommendations. Inspired by the intuition that humans usually reflect and learn from mistakes, in this paper, we attempt to build a self-correction learning loop (dubbed ReLoop) for recommender systems. In particular, a new customized loss is employed to encourage every new model version to reduce prediction errors over the previous model version during training. Our ReLoop learning framework enables a continual self-correction process in the long run and thus is expected to obtain better performance over existing training strategies. Both offline experiments and an online A/B test have been conducted to validate the effectiveness of ReLoop.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Recommender System, CTR prediction, Self-Correction

[†] Both authors contributed equally to this work.
[*] Corresponding authors.

---

1 INTRODUCTION

Nowadays, recommender systems play an indispensable role in our daily life. With the ever-growing volume of online information, recommender systems have been widely adopted to help users discover items of their interests in various applications, such as e-commerce, news feeds, music apps, movie websites, and so on. The quality of recommendation could not only influences user experience, but also has a direct impact on platform revenues. As a consequence, the research of recommender systems has received much attention from both academia and industry in recent years [30].

Model deployment in an industrial recommender system usually comprises two interleaved processes: offline (incremental) training and online inference. The deployed model needs to undergo frequent re-training to capture time-varying user behaviors from newly collected interaction logs. This forms the conventional learning loop: *model training→online inference→feedback data→model training → ⋯*. First, after offline training, the model is deployed online to serve user requests. Then, new feedback data are collected as input (e.g., clicked user-item pairs as positive samples and unclicked ones as negative samples) for the next round of model training, which in turn delivers a new model version for online inference. Such a training process only acquires users' feedbacks (e.g., clicks) as supervision signals, but dose not take into account the errors made in previous model versions. This wastes the opportunity of accumulating knowledge from the past recommendation experience for self-correction.

As an analogy, humans are taught to learn from mistakes. Taking class learning for example, students need to review their past wrong question collections regularly to achieve better grades in the next exam. This inspires us to think about how to reflect and learn from the past errors for recommender systems, which is an interesting yet unexplored research problem. Existing studies focus mostly on designing better models to learn from the historical data, e.g.,

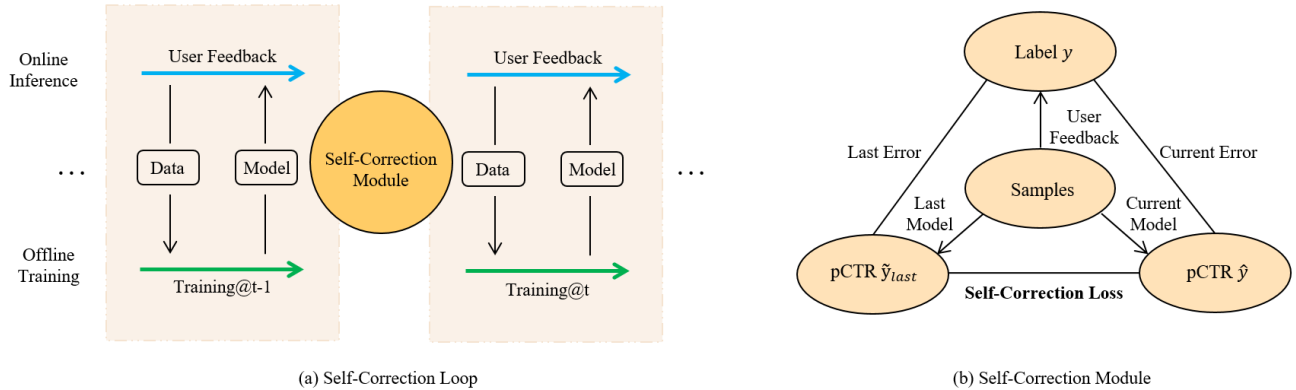(a) Self-Correction Loop   (b) Self-Correction Module

Figure 1: (a) illustrate two sequential training procedures, which are connected by the self-correction module; (b) details the self-correction module, where errors are defined from model's prediction score and users' feedback label. We expect the training process could reduce these errors for better online serving.

via feature interactions [4, 25], user behavior modeling [18, 31, 32] and multi-task learning [14, 15]. The most related work in this direction includes knowledge distillation [10, 29] and continual learning [17]. Knowledge distillation could be applied to mimicing the behaviors of previous models, while continual learning techniques aim to address the catastrophic forgetting issue during continuous model training. Yet, neither of them focuses on developing the self-correction ability of models that could continuously reflect on their past errors and improve themselves.

In this work, we propose a novel learning framework named ReLoop, with the goal of building a self-correction learning loop for recommender systems. There is a large design space to explore for our ReLoop framework: for example, using sample-wise testing errors, list-wise ranking errors, or even online business metrics (e.g., clicks, revenues) of a recommendation list as the error measure. As an initial attempt in this work, we opt for a simple yet effective design choice and demonstrate its use in the click-through rate (CTR) prediction task. In the training loop of recommender systems, data samples are continuously collected from the deployed model online, and thus we could obtain both the click labels from users and the corresponding prediction errors from the previous models. Our key insight is that a new model should be forced to make smaller prediction errors than the previous model versions on the same data samples, and thus achieves the ability of self correction. With this insight, we choose sample-wise prediction errors (i.e., gaps between prediction scores and ground truths) as the error measure and further propose a customized loss function to force every new model version to reduce prediction errors over the previous model version during training. Our ReLoop framework enables a continual self-correction process in the long run and thus is expected to obtain better performance than existing training strategies. The framework is generic and model-agnostic because it is applicable to the training of any existing recommendation model.

To validate the effectiveness of ReLoop, we conduct extensive experiments on both public benchmark datasets and a private industrial dataset. The results from both offline experimental results and

an online A/B test show consistent improvements over the baselines when applying our ReLoop learning framework. The main contributions of this paper are summarized as follows:

- We identify the importance of reflecting on past errors, and propose a generic ReLoop learning framework for continuous self-correction in recommender systems.
- We design a simple and effective self-correction module based on a customized loss, which requires nearly no engineering work in the online serving system.
- Both offline experiments and an online A/B test have been conducted to demonstrate the simplicity and effectiveness of our ReLoop framework.

2 RELOOP FRAMEWORK

In this section, we first describe the training loop in industrial recommender systems. Then we define the errors made in the loop, based on which a self-correction module is introduced. Finally, we integrate the self-correction module into the CTR prediction learning paradigm and compare it with the knowledge distillation method.

2.1 Training Loop In Production

The training loop of a recommendation model is illustrated in Figure 1(a). Firstly, training data is collected from user implicit feedback on the exposed items, i.e., clicked items as positive samples and unclicked items as negative samples. Secondly, a ranking model is trained using a sliding window of such data (w.r.t. both offline/incremental training modes). Then, the new model will be updated to the online serving worker. During the online inference stage, candidates are evaluated by the model and displayed to users according to the ranking of prediction scores. Finally, online exposure and click events will be recorded in the user behavior log, which in turn triggers a new period of training. This forms the training loop for recommender systems. Nevertheless, we can see that the connection between adjacent training procedures are very loose, since each one trains its own model independently. In other words, current training procedure does not fully utilize information

from previous procedure, and ignores the errors made in previous predictions. To alleviate this problem, we introduce a novel self-correction module to learn from previous mistakes.

2.2 Self-Correction Module

Figure 1(b) presents the design details of the proposed self-correction module. Each sample has a label y, which is generated from users' implicit feedback (e.g., clicks). The predicted click-through rate (pCTR) could be recorded during model inference, which represents the probability of a user clicking on an item. Intuitively, we define the gap between a sample's pCTR and its ground truth label as the error made, so we could get the last error from the last pCTR $\widetilde{y}_{last}$ for each sample. As for the current error, it denotes the gap between the current prediction $\hat{y}$ and the ground truth label during training. Inspired by the fact that human beings learn from mistakes, we expect the model at time t performs better than the last $t-1$ version. For the purpose of building such an evolving system, we design a novel self-correction loss, which forces the training process get lower error than the previous one. For clarity, we take a positive sample ($y = 1$) for example:

$$y - \hat{y} \leq y - \widetilde{y}_{last} \quad \Leftrightarrow \quad \widetilde{y}_{last} - \hat{y} \leq 0 \tag{1}$$

where y is the label, $\hat{y}$ is the predicted CTR in current training step, and $\widetilde{y}_{last}$ is the predicted CTR from last version of the model. In order to reduce the error from the past and guide the training process to a better direction, we propose to penalize those samples on which it performs worse than the last version using the following loss:

$$\mathcal{L}_{sc} = y \max(\widetilde{y}_{last} - \hat{y}, 0) + (1 - y) \max(\hat{y} - \widetilde{y}_{last}, 0) . \tag{2}$$

The loss $\mathcal{L}_{sc}$ is an extension from the widely used hinge loss. Note that the last pCTR $\widetilde{y}_{last}$ could be obtained from previous model, or directly collected from the online inference server. In this way, samples incorrectly predicted by the previous model could be emphasized when the current model does not behavior better than the previous version.

2.3 Training Strategy

The common objective function for CTR prediction task is the binary cross-entropy loss as follows:

$$\mathcal{L}_{ce} = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) \tag{3}$$

Then our ReLoop loss function is formulated as a sum of both terms:

$$\mathcal{L} = \alpha \mathcal{L}_{sc} + (1 - \alpha) \mathcal{L}_{ce} \tag{4}$$

where $\alpha \in [0, 1]$ is the hyper-parameter to adjust the importance of self-correction loss and cross-entropy loss. Thanks to the design for reflecting on past errors, our proposed ReLoop framework enables the CTR prediction task to be optimized in a self-correction manner.

2.4 Comparison with Knowledge Distillation

Our design bears a similarity to the technique of knowledge distillation [7], which could be formulated as follows:

$$\mathcal{L}_{kd} = -\widetilde{y}_{last} \log \hat{y} - (1 - \widetilde{y}_{last}) \log(1 - \hat{y}) \tag{5}$$

The main difference is that knowledge distillation is designed to mimic the behaviors of a teacher model, while our self-correction

learning module is proposed to learn from past errors of the previous model. It is worth noting that teachers could make mistakes too. Take Figure 2 (a) for example, given a positive sample, the predicted score from the previous model is $\widetilde{y}_{last} = 0.8$. If current prediction $\hat{y}$ is lower than $\widetilde{y}_{last}$, namely performing worse than the previous version, the basic $\mathcal{L}_{ce}$ loss will be assisted by the self-correction loss $\mathcal{L}_{sc}$. On the other hand, if $\hat{y}$ has already been higher than $\widetilde{y}_{last}$, namely performing better than before, we do not impose extra penalty on the model.

(a) $y = 1, \widetilde{y}_{last} = 0.8$ (b) $y = 0, \widetilde{y}_{last} = 0.3$
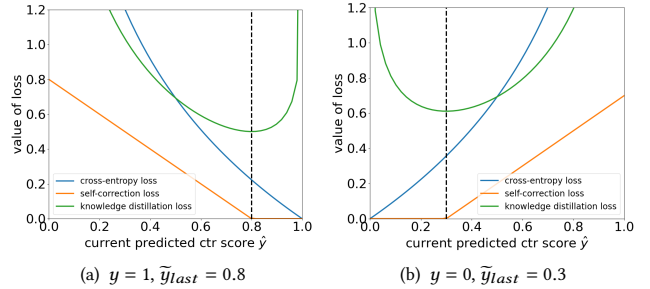
Figure 2: The Loss functions of cross-entropy loss, knowledge distillation loss and our proposed self-correction loss.

3 EXPERIMENTS

In this section, we first conduct extensive offline experiments on four public datasets. Then, we deploy our ReLoop framework in an application of one industrial Newsfeed product, where hundreds of millions of users browse news and videos everyday. We aim to answer the following research questions:

RQ1: How does our proposed method ReLoop perform against the state-of-the-art methods?

RQ2: How does the hyper-parameter influence the performance?

RQ3: How dose ReLoop perform in online A/B tests?

3.1 Evaluation on Public Datasets

3.1.1 Experiment Setup. We conduct experiments on four public datasets following previous works [2, 5, 20], and on a large-scale industrial dataset.

- **Criteo** is a famous benchmarking dataset for CTR prediction tasks. It contains 45M instances, and each instance consists of 13 numerical feature fields and 26 categorical fields.
- **Avazu** is a mobile advertisements dataset. It contains 40M instances with 22 feature fields including advertisement attributes and user features.
- **MovieLens** dataset consists of users' tagging records on movies. It contains 2M instances and focus on the personalized tag recommendation.
- **Frappe** dataset is collected from a context-aware app discovery tool. It contains 288k instances and 10 fields for each input.
- **Production** dataset is consisted of more than 500M instances and each one has over 100 feature fields, sampled from an industrial news feed product.

For the public datasets, we split the instances by 8:1:1 for training, validation and testing, following the experiments setting by [2].

Since these public datasets do not contain historical user behavior sequences, user behavior based models are not suitable for comparison. To get last prediction score for public datasets, we use 90% of training samples, mimicking the online setting, to train the previous model version and then infer on the whole training set to generate $\widetilde{y}_{last}$ values .

Table 1: Comparison of competing methods and ReLoop on Four public datasets.

Models	Criteo AUC	Criteo Logloss	Avazu AUC	Avazu Logloss	MovieLens AUC	MovieLens Logloss	Frappe AUC	Frappe Logloss
LR	0.7858	0.4636	0.7313	0.4065	0.9215	0.3080	0.9329	0.2860
FM	0.7933	0.4574	0.7496	0.3740	0.9388	0.2797	0.9641	0.2143
NFM	0.7968	0.4537	0.7531	0.3761	0.9441	0.3004	0.9727	0.2079
IPNN	0.8026	0.4509	0.7526	0.3737	0.9469	0.2792	0.9735	0.2012
Wide&Deep	0.8062	0.4453	0.7529	0.3744	0.9381	0.3310	0.9728	0.2038
DCN	0.8059	0.4463	0.7550	0.3721	0.9419	0.2791	0.9402	0.2808
DeepFM	0.8025	0.4501	0.7535	0.3742	0.9424	0.3131	0.9719	0.2108
xDeepFM	0.8070	0.4443	0.7535	0.3737	0.9448	0.2717	0.9738	0.2098
FmFM	0.8056	0.4462	0.7603	<u>0.3685</u>	0.9465	0.2714	0.9749	0.2004
AFN+	0.8074	0.4451	0.7555	0.3718	0.9500	0.2585	0.9783	0.1762
KD+DCN	<u>0.8136</u>	<u>0.4473</u>	<u>0.7653</u>	0.3906	0.9648	<u>0.2164</u>	<u>0.9837</u>	0.1672
KD+DeepFM	0.8135	0.4462	0.7652	0.3690	<u>0.9650</u>	0.2183	0.9803	<u>0.1552</u>
RLP+DCN	0.8138	**0.4381**	0.7648	**0.3667**	**0.9693**	0.2189	0.9843	0.1660
RLP+DeepFM	**0.8139**	0.4382	**0.7659**	0.3750	0.9687	**0.2160**	**0.9848**	**0.1549**

Baselines. We compare our ReLoop model with many state-of-the-art approaches including LR [16], FM [21], NFM [6], PNN [19], Deep&Cross [22], Wide&Deep [1], DeepFM [4], xDeepFM [11], FmFM [24], AFN+ [2], Distill [7].

3.1.2 Experiment results. The experiments results are shown in Table 1. We take Area under ROC curve (AUC) and Logloss as offline metrics. We utilize two representative methods, namely DCN and DeepFM as backbone for our framework. From this table, we can see that methods based on ReLoop outperform all the other baselines by a large margin, demonstrating the superiority and generalization to different datasets. ReLoop can be regarded as plug-in component for improving CTR models performance. Compared to original backbone models, our framework can polish up consistently and significantly. Specifically, It achieves AUC 1.42%, 1.65%, 2.91% and 4.69% improvement relative to the backbone models, which prove the effectiveness of our novel self-correction module. In addition, knowledge distillation technique can also promote the accuracy and become the best baseline on almost all datasets.

3.2 Evaluation in Production

Inspired by the promising improvement of ReLoop framework on public datasets, we deploy our method to a real world NewsFeeds recommender system. Firstly, the offline experiments are conducted, and results are shown in Table 2. We can see that ReLoop outperform baseline and knowledge distillation method on both AUC and Logloss metrics. It is worth noting that our framework introduces little cost on training and no cost on inference, making it convenient to be deployed on existing system. Besides, we conduct sensitive experiments in terms of hyper-parameter α from equation

4, shown in the Figure 3. The purple line is the baseline method which does not adopt the self-correction loss and we can see that ReLoop framework perform better consistently.

Table 2: Comparison between knowledge distillation and ReLoop on production dataset.

	AUC	Improv.	Logloss	Improv.	Time cost
Baseline	0.7394	-	0.2959	-	-
KD+Baseline	0.7458	+0.87%	0.2743	+7.30%	+1.8%
RLP+Baseline	0.7483	+1.20%	0.2313	+21.83%	+1.8%

Table 3: The improvement of our ReLoop method compared to baseline in one-week online A/B tests.

Day 1	Day 2	Day 3	Day 4
+1.38%	+2.01%	+1.34%	+0.64%

Day 5	Day 6	Day 7	Average
+2.61%	+1.13%	+1.09%	+1.46%

For online A/B tests, we split the users into two groups, each of which has more than 300 thousands activate users. The first group receives recommendation results generated by a prediction model without ReLoop approach, which has been used by the current product; the second one receives the recommended news by the same backbone with ReLoop approach. We firstly conduct one week A/A tests to record the minor difference between two user groups and take CTR as the evaluation metric. During one week online A/B tests, our ReLoop method is consistently better than the baseline model, where CTR improvement achieves about 1.46% on average (A/B - AA), detailed improvements are shown in Table 3, which verify the effectiveness of our method. Deploying ReLoop to an existing CTR model is convenient and requires nearly no engineering work in the online serving system.
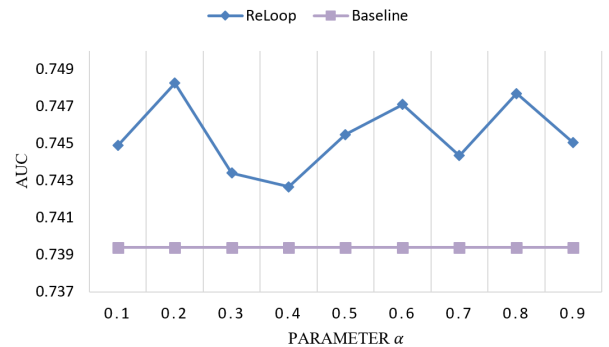

Figure 3: The comparison between baseline and ReLoop on different value of parameter α in terms of AUC metric.

4 RELATED WORK

In this section, we review the related work from three aspects: *CTR prediction*, *continual learning*, and *knowledge distillation.*

CTR Prediction. The CTR (click-through rate) prediction task enables the ranking of a small set of candidate items by modeling fine-grained features of users, items and contexts. This task usually involves large-scale highly sparse data, which consist of multi-fields of numerical or categorical features. Due to the huge research and economic value of this task, many research efforts have been made in this direction. Typical models of CTR prediction range from simple logistic regression (LR) models [16], factorization machines (FM) [21], to deep neural networks (DNN) [1, 4]. Readers could refer to the recent benchmarking work on CTR prediction and recommendation [34, 35] for a more comprehensive view.

In summary, current research on CTR prediction focuses mostly on the design of effective model structures, including feature interactions [4, 25], sequential user behaviour modeling [13, 26, 31, 32], multi-task learning [14, 15], multi-scenario prediction [23], automated architecture search [12], and so on. Nevertheless, the research community focuses less on the learning process in a continuous training loop, which is equally important in practice. Our work optimizes the learning loop through self-correction from past errors.

Continual Learning. Continual learning [17], as a lifelong learning concept, has received much attention in recent years. In a typical setting of continual learning, a model is set to learn different tasks sequentially without forgetting knowledge obtained from the preceding tasks. Therefore, most research focuses on mitigating the catastrophic forgetting issue and maximally retaining its accuracy on the old tasks that have been trained. Instead, in the continual learning process of recommender systems (e.g., [27]), the model is periodically trained on the same task with new data that may have distribution shift to a certain degree. Meanwhile, we only care about the model performance on the new data. Yet, the similarity lies in that our work also intends to retain the past knowledge in the continuous training process for model improvement. This new continual learning setting in recommender systems may deserve more attention and exploration. A concurrent work [8] also applies continual learning for CTR prediction. In contrast, we focus on the self-correction ability in continual learning.

Knowledge Distillation. The idea of knowledge distillation (KD) [7], which aims to distill the dark knowledge from a teacher model to a student model, has been widely applied in model compression and knowledge transfer [3]. Some recent work also applies KD to recommendation tasks, e.g., CTR prediction [33], topology distillation [9], and multi-task transfer [28]. Instead, our work is mostly related to self-distillation [10, 29], which proposes to improve the generability of a model through previous model checkpoints or predictions. They regard the previous model as a teacher and take the prediction scores from teacher as soft labels to guide the current training process. However, *teachers are not always correct*, and these methods fail to learn from the mistakes made in previous models. We make a distinction between our ReLoop loss and the KD loss in Figure 2.

5 CONCLUSION AND FUTURE WORK

In this paper, we propose a generic learning framework, named ReLoop, for recommender systems, which can build a self-correction learning loop. Inspired by the learning paradigm of human beings, we introduce a novel self-correction module that takes into account the errors made in previous recommendations. In order to actively reflect on past errors, we design a customized loss that penalizes those samples on which the model performs worse than the past. Meanwhile, the proposed ReLoop framework applies to any existing model in recommender systems. Extensive experiments on four popular public datasets and one large-scale real industrial dataset have been conducted, and the performance results show the superiority of our framework than other state-of-the-art methods. In addition, we have deployed the framework on an industrial Newsfeed scenario and the online A/B test demonstrate the effectiveness of ReLoop method. In addition, there are some potential directions for future exploration. Examples include more fine-grained definition of errors, such as list-wise utility or session-based revenue, and more effective implementation (beyond loss) of the self-correction module. We hope our work could inspire more research in this direction.

6 ACKNOWLEDGEMENT

REFERENCES

[1] 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.

[2] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive Factorization Network: Learning Adaptive-Order Feature Interactions. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 3609–3616.

[3] Jianping Gou, Baosheng Yu, Stephen John Maybank, and Dacheng Tao. 2020. Knowledge Distillation: A Survey. *CoRR* abs/2006.05525 (2020).

[4] 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.

[5] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

[6] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. 355–364.

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

[8] Ke Hu, Yi Qi, Jianqiang Huang, Jia Cheng, and Jun Lei. 2022. Continual Learning for CTR Prediction: A Hybrid Approach. *CoRR* abs/2201.06886 (2022).

[9] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2021. Topology Distillation for Recommender System. In *The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 829–839.

[10] Kyungyul Kim, Byeongmoon Ji, Doyoung Yoon, and Sangheum Hwang. 2020. Self-Knowledge Distillation: A Simple Way for Better Generalization. *CoRR* abs/2006.12000 (2020).

[11] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, et al. 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, (KDD)*. 1754–1763.

[12] Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. 2020. AutoFIS: Automatic Feature Interaction Selection in Factorization Models for Click-Through Rate Prediction. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. 2636–2645.

[13] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. 2021. Noninvasive Self-attention for Side Information Fusion in Sequential

Recommendation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI)*. 4249–4256.

[14] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. 2018. Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1930–1939.

[15] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *Proceedings of the 41st International Conference on Research & Development in Information Retrieval (SIGIR)*. 1137–1140.

[16] H. Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 1222–1230.

[17] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks* 113 (2019), 54–71.

[18] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User Behavior Retrieval for Click-Through Rate Prediction. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*. 2347–2356.

[19] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-Based Neural Networks for User Response Prediction. In *Proceedings of the IEEE 16th International Conference on Data Mining (ICDM)*. 1149–1154.

[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. *ACM Trans. Inf. Syst.* 37, 1 (2019), 5:1–5:35.

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

[22] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 255–262.

[23] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, and Xiaoqiang Zhu. 2021. One Model to Serve All: Star Topology Adaptive Recommender for Multi-Domain CTR Prediction. In *The 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 4104–4113.

[24] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed Factorization Machines for Recommender Systems. In *Proceedings of the Web Conference (WWW)*. 2828–2837.

[25] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference (WWW)*. 1785–1797.

[26] Yikai Wang, Liang Zhang, Quanyu Dai, Fuchun Sun, Bo Zhang, Yang He, Weipeng Yan, and Yongjun Bao. 2019. Regularized Adversarial Sampling and Deep Time-aware Attention for Click-Through Rate Prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*. 349–358.

[27] Minhui Xie, Kai Ren, Youyou Lu, Guangxu Yang, Qingxing Xu, Bihai Wu, Jiazhen Lin, Hongbo Ao, Wanhong Xu, and Jiwu Shu. 2020. Kraken: memory-efficient continual learning for large-scale real-time recommendations. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. 21.

[28] Chenxiao Yang, Junwei Pan, Xiaofeng Gao, Tingyu Jiang, Dapeng Liu, and Guihai Chen. 2022. Cross-Task Knowledge Distillation in Multi-Task Recommendation. *CoRR* (2022).

[29] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. [n.d.]. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*. 3712–3721.

[30] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.

[31] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2018. Deep Interest Evolution Network for Click-Through Rate Prediction. *CoRR* abs/1809.03672 (2018).

[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 SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*. 1059–1068.

[33] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM)*. 2941–2958.

[34] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *The 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 2759–2769.

[35] Jieming Zhu, Kelong Mao, Quanyu Dai, Liangcai Su ands Rong Ma, Jinyang Liu, Guohao Cai, Zhicheng Dou, Xi Xiao, and Rui Zhang. 2022. BARS: Towards Open Benchmarking for Recommender Systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.