Multi-Branch Cooperation Networks for Enhanced Click-Through Rate Prediction in Large-Scale E-Commerce Search

Xu Chen*
Taobao & Tmall Group of Alibaba
Hangzhou, China
huaisong.cx@alibaba-inc.com

Zida Cheng
Taobao & Tmall Group of Alibaba
Hangzhou, China
chengzida.czd@alibaba-inc.com

Shuai Xiao
Taobao & Tmall Group of Alibaba
Hangzhou, China
shuai.xsh@alibaba-inc.com

Chen Ju
Taobao & Tmall Group of Alibaba
Hangzhou, China
juchen.ju@alibaba-inc.com

Xiaoming Liu
Xi'an Jiaotong University
Xi'an, China
xm.liu@xjtu.edu.cn

Jinsong Lan
Taobao & Tmall Group of Alibaba
Beijing, China
jinsonglan.ljs@alibaba-inc.com

Xiaoyong Zhu
Taobao & Tmall Group of Alibaba
Hangzhou, China
xiaoyong.z@alibaba-inc.com

Bo Zheng
Taobao & Tmall Group of Alibaba
Hangzhou, China
bozheng@alibaba-inc.com

Abstract

Existing Click-Through Rate (CTR) prediction models use various feature interaction techniques, each with unique strengths, but relying on a single type limits their ability to capture complex relationships. Recent research shows that effective CTR models often combine an MLP network with a dedicated feature interaction network in a two-parallel structure. However, the interplay and cooperative dynamics between different streams or branches remain under-researched. In this work, we introduce a novel Multi-Branch Cooperation Network (MBCnet) which enables multiple branch networks to collaborate with each other for better complex feature interaction modeling. Specifically, MBCnet consists of three branches: the Extensible Feature Grouping and Crossing (EFGC) branch that promotes the model's memorization ability of specific feature combinations, the low rank Cross Net branch and Deep branch to enhance explicit and implicit feature crossing for generalization. Among them, a novel cooperation scheme is proposed based on two formulated objectives: ***branch co-teaching*** that encourages well-learned branches to support poorly-learned ones on specific training samples, and ***moderate differentiation*** that advocates branches to maintain a reasonable level of difference in their feature representations on the same inputs. This cooperation strategy improves learning through mutual knowledge sharing and boosts the discovery of diverse feature interactions across branches. Extensive experiments on large-scale industrial datasets and online A/B test demonstrate MBCnet's superior performance.

*corresponding author

WWW Companion '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2308-7/2026/04
https://doi.org/10.1145/3774905.3795839

CCS Concepts

• **Information systems → Retrieval models and ranking**.

Keywords

multi-branch cooperation, CTR prediction, feature grouping, branch co-teaching, moderate differentiation

ACM Reference Format:
Xu Chen, Zida Cheng, Shuai Xiao, Chen Ju, Xiaoming Liu, Jinsong Lan, Xiaoyong Zhu, and Bo Zheng. 2026. Multi-Branch Cooperation Networks for Enhanced Click-Through Rate Prediction in Large-Scale E-Commerce Search. In *Companion Proceedings of the ACM Web Conference 2026 (WWW Companion '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3774905.3795839

1 Introduction

Click-Through Rate (CTR) prediction which estimates the probability of a user clicking on a candidate item, is a fundamental task in online services like recommendation, retrieval, and advertising [2, 4, 20, 35]. One key aspect of developing precise CTR prediction is to capture complex feature interactions.

For many years, researchers have examined the role of feature interaction in CTR by employing a range of methodologies. Earliest algorithm is Logistic Regression (LR) [10], which is a linear model and heavily relies on hand-crafted input features by domain experts. Due to the superior non-linear feature learning ability of Deep Neural Networks (DNN), many scholars have investigated deep learning techniques on CTR prediction [30]. For example, Wide & Deep [4] consists of a jointly trained wide linear model and deep neural networks, combining the benefits of memorization and generalization for recommender systems. DeepFM [6] and xDeepFM [13] combine the power of factorization machines and deep learning, to emphasize both low- and high-order feature interactions. Instead of employing FM, Wang et al. [27, 29] introduced a novel cross network that explicitly performs feature crossing with each layer, which eliminates the need for manual feature engineering. Mao et al. [17] developed a simple yet strong dual MLP model

to capture diverse feature interactions and this model has achieved state-of-the-art performance on public benchmarks.

Each feature interaction technique brings its own advantages, and solely depending on one may hinder the model's potential to capture complex feature relationships. Particularly in industrial contexts with vast numbers of users and items, the data patterns can be exceedingly intricate. We usually incorporate hundreds of input feature fields to learn for the data patterns, but such many feature fields also exhibit enormous useful feature interactions. The modeling ability of single feature interaction technique is limited and insufficiently effective [15]. Relying on one single approach cannot well cover the complex high-order feature space for CTR prediction. Recent researchers have highlighted that existing successful CTR models [4, 6, 13, 23, 27, 29] usually adopt a two-branch architecture and work in an ensemble style. The ensemble style permits the model to learn feature interactions from different perspectives. *Despite the success of above models, the interplay and cooperative dynamics between different streams or branches remain under-researched.* Recent models ensemble different techniques for CTR prediction by prediction voting or latent feature mean pooling or concatenation, which means no explicit interplay and cooperative signals are provided. It could largely limit the whole model's learning ability, especially when majority branches have wrong predictions. It motivates us to study the working principles of multi-branch CTR networks and develop a more effective model to capture the complex patterns.

In this work, we propose a novel Multi-Branch Cooperation Network (MBCnet) which enables multiple branch networks to collaborate with each other for better complex feature interaction modeling. MBCnet consists of three different prediction branches and a cooperation scheme. One important prediction branch is a designed Extensible Feature Grouping and Crossing (EFGC) module, which groups hundreds of feature fields and only conducts feature interaction between specific group pairs by domain-driven knowledge. This branch enhances the model's memorization ability to capture domain-driven feature interactions. The other two branches are the existing popular and powerful low-rank Cross Net [29] for explicit feature crossing and Deep Net [6, 27] for implicit feature crossing, which both improve the model's generalization ability [4]. More importantly, we propose a multi-branch cooperation scheme based on two principles to better exert advantages of different branches. *The first principle (**branch co-teaching**) dictates that well-learned branch should assist poorly-learned branch on particular samples, while the second principle (**moderate differentiation**) maintains that latent features of distinct branches should preserve a moderate level of differentiation on the same input samples, avoiding extremes of either excessive divergence or excessive similarity.* Specifically, the first principle is embodied in a co-teaching objective function, where the well-learned branch uses its predictions as pseudo labels to guide the poorly-learned one on disagreed predicted samples. The second principle is formulated as an equivalent transformation regularization loss between latent features of branches. This approach promotes learning abilities of different branches through knowledge transfer and also enhances their capabilities to uncover various patterns of feature interactions. Through experiments, we have shown that MBCnet can largely improve CTR prediction performance on our industrial datasets, and has achieved obvious online performance with an absolute **0.09 point** CTR improvement, a relative **1.49%** deal growth and a relative **1.62%** Gross Merchandise Value (GMV) rise. The contributions are summarized as follows:

- We propose a novel multi-branch cooperation network (MBCnet) which ensembles different feature interaction branches to enlarge the capacity of capturing complex feature patterns in industrial data.
- We introduce a novel cooperation scheme that facilitates cooperation and complements of branches to promote the overall learning ability. The cooperation scheme is constructed upon two principles, which promote each branch's learning ability and meanwhile enabling them to uncover distinct patterns even with the same inputs.
- Through extensive experiments on our large-scale datasets, we demonstrate the effectiveness of the proposed method. MBCnet has been deployed in image2product retrieval at our e-commerce app, and achieved obvious improvements .

2 Related Work

Click-Through Rate (CTR) Prediction: Capturing the complex feature interactions is a key to a successful CTR model. Early works combine Logistic Regression (LR) [10] and Factorization Machine (FM) [22] to capture feature interactions. While these methods either rely much on handcrafted features or focus on lower-order feature interactions, showing limitations in learning complex feature patterns [32]. With the superior capacity of deep neural networks (DNN) for feature extraction, deep learning techniques have been widely examined in the context of CTR prediction [4, 6, 28]. Wang et al. [29] designed Deep and Cross Network (DCN) which can more efficiently and explicitly learn certain bounded-degree feature interactions. They later introduced a mixture of low-rank expert network into DCN to make the model more practical in large-scale industrial settings [27]. In FinalMLP [17], researchers observed that even two parallel MLP networks can achieve satisfied performance. They further proposed feature gating and interaction aggregation layers that can be easily plugged to make an enhanced two-stream MLP model. Some other CTR works investigate sequential user behavior modeling [20, 34, 35], cross-domain knowledge transfer [1–3, 11, 21, 24], multi-task learning [16, 25] and current popular large language models (LLMs) augmented CTR methods [5, 9, 12, 14, 19] for improved performance. These topics are beyond the scope of this paper, so we do not elaborate them here.

Ensemble Learning: Ensemble learning is a machine learning technique that aggregates two or more learners (*e.g.*, regression models, neural networks) to produce better prediction performance [18, 31]. Generally speaking, an ensemble learner is more robust and able to perform better than the individual learners. In CTR prediction, different feature interaction techniques exhibit different modeling advantages. It is necessary to combine them together to better capture the complex patterns. Researchers have investigated the ensemble idea in CTR prediction from several aspects [7, 15]. Ling et al. [15] investigated eight ensemble variants of GBDT, LR and DNN. They found that initializing the GBDT sample target with DNN's prediction score yielded the best performance in their business context. These methods do not work in an end-to-end style and require hard efforts on hand-craft tuning. Deep learning based ensemble CTR works [4, 6, 28, 36] are becoming more popular. For instance, EnKD [36] specifically studies knowledge

Table 1: The designed feature groups of EFGC branch in our image2product search.

group	feature fields	intention of feature interactions
1	*query image, item image*	visual relevance between query and item
2	*query image, item attributes*	relevance between visual query and item attributes
3	*user profile, item image*	user preference on visual patterns of images
4	*user profile, item attributes*	generalized user preference on item attributes
5	*userID, itemID*	personalized user preference on items
6	*queryClusterID, docClusterID*	matching degree between clustered query and item images
7	*queryClusterID attributes, docClusterID attributes*	non-visual relevance between query and item

distillation technique for model ensemble in order to learn light and accurate student prediction model. Compared to recent ensemble works [4, 17, 28, 36], our model explicitly advocates branches to learn from each other and uncover different feature patterns even under the same supervision.

3 Method

3.1 Overview

In CTR prediction, given the raw input feature $\boldsymbol{x} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_M}\}$ with M feature fields and user feedback label $\boldsymbol{y} \in \{0, 1\}$, we aim to learn a mapping function $\mathcal{F} : \boldsymbol{x} \rightarrow \boldsymbol{y}$ to predict user behaviors for online services. Especially in industrial scenarios, the number of feature fields M can be in hundreds or even thousands, bringing great challenges of modeling feature interactions. MBCnet comprises three feature interaction branches and an innovative multi-branch cooperation scheme. The general architecture is given in Figure 1.

3.2 Extensible Feature Grouping and Crossing

The EFGC branch is designed to improve the model's memorization capability for intended interactions of specific feature fields, guided by domain-driven knowledge. It consists of two main components: the embedding layer, and the feature grouping and crossing module.

Embedding Layer: The embedding layer aims to encode raw input feature $\boldsymbol{x} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_M}\}$ as embedding vectors. Each field has specific meaning such as *userID*, *age* and *gender*. Let E_i be the i-th embedding function of $\boldsymbol{x_i}$, the embedding layer is written as:

$$\{\boldsymbol{e_1}, \boldsymbol{e_2}, ..., \boldsymbol{e_M}\} = \{E_1(\boldsymbol{x_1}), E_2(\boldsymbol{x_2}), ..., E_M(\boldsymbol{x_M})\} \tag{1}$$

where $\boldsymbol{e_i}$ is the embedding vector of $\boldsymbol{x_i}$.

Feature Grouping: In CTR prediction, different feature fields have different physical meanings, and their interactions are crucial for personalized ranking. For instance, in e-commence search, combining user profile feature fields (*e.g., userID, age*) and item profile feature fields (*e.g., itemID, price, sales volume*) can reveal a user's general preferences regarding item price and quality. Conversely, interactions like *item title* and *item category* could show less contributions to personalized ranking. *In essence, different feature field interactions have varying degrees of influence on CTR prediction.* The common practice of concatenating $\boldsymbol{e_i}$ in Eq. 1 may fail to highlight certain specific interactions and introduce irrelevant feature patterns into subsequent deep modules.

It is crucial to organize feature fields into distinct groups and perform feature crossing within these specific groups, driven by domain knowledge. Specifically, we follow three main steps: 1) *Identify Desired Matching Patterns*: Determine the types of matching patterns that need to be emphasized based on the problem domain. 2) *Group Feature Fields*: Select and group the feature fields in a way that aligns with the identified matching patterns. 3) *Integrate Group Embeddings into Subsequent Layers*: Input the embeddings from each group into the following layers to facilitate the learning of feature interactions. For instance, in step 1 of our image2product search, we aim to emphasize the visual relevance between query image and item image, while personalizing the results based on each user's preferences on item attributes. Step 2 is formulated as:

$$\boldsymbol{e_i}^{\text{Group}} = \boldsymbol{e_x} \oplus, ..., \oplus \boldsymbol{e_y} \tag{2}$$

where $\boldsymbol{e_i}^{\text{Group}}$ denotes i-th group that combines embeddings of specific feature fields. $\oplus$ is the concatenation operation. Detailed designed groups in our image2product search are listed in Table 1. In this table, query and item image are encoded as vectors by deep networks. The *user profile* includes fields such as *userID, age, job, gender, consumption level* and other personalized profiles. The *item attributes* encompasses fields like *itemID, price, sales volume, delivery location, brand* and other relevant information. The *queryClusterID* and *docClusterID* mean the clustering ID of query images and item images. The *queryClusterID attributes* includes features such as *query image category*, *average price of query related items*.

Feature Crossing: After feature grouping, we can perform feature crossing with subsequent MLP layers as follows:

$$\boldsymbol{h_i}^{\text{EFGC}} = f_i^{\text{EFGC}}(\boldsymbol{e_i}^{\text{Group}}) \tag{3}$$

where $\boldsymbol{h_i}^{\text{EFGC}}$ denotes the output of i-th group feature crossing. f_i^{EFGC} is a non-linear MLP network. Finally, as shown in Figure 1, the output of our EFGC branch, denoted as $\boldsymbol{h}^{\text{EFGC}}$, is formulated as:

$$\boldsymbol{h}^{\text{EFGC}} = \mathbb{FC}(\boldsymbol{h_i}^{\text{EFGC}} \oplus, ..., \oplus \boldsymbol{h_{N_g}}^{\text{EFGC}}) \tag{4}$$

where N_g is the number of feature groups and $\mathbb{FC}$ means one fully-connected layer to reduce the output dimension.

3.3 Deep Net and Low Rank Cross Net

Deep Net: Deep neural networks, composed of stacked non-linear MLP layers, have the capability to implicitly learn feature interactions [4, 6, 13, 17, 27, 29]. The input of Deep Net is typically formed by concatenating all embedding fields as:

$$\boldsymbol{e}^{\text{Deep}} = \boldsymbol{e_1} \oplus \boldsymbol{e_2} \oplus, ..., \oplus \boldsymbol{e_M} \tag{5}$$

Let f^{Deep} be the stacked non-linear MLP layers of deep Net branch. The output of Deep Net branch, denoted as $\boldsymbol{h}^{\text{Deep}}$, is written as:

$$\boldsymbol{h}^{\text{Deep}} = f^{\text{Deep}}(\boldsymbol{e}^{\text{Deep}}) \tag{6}$$

Low Rank Cross Net: Despite the effectiveness of Deep Net in modern applications, it often struggles with capturing feature interactions, especially higher-order ones [29]. The popular low rank cross net(*i.e.*, CrossNetV2) maps feature interactions in low-rank space and employs a mixture of experts architecture to improve its expressiveness. The input of low-rank cross net branch is also the concatenation of all field embeddings, which means $\boldsymbol{e}^{\text{Cross}} = \boldsymbol{e}^{\text{Deep}}$. Then the resulting low-rank cross net layer is formulated as:

$$\boldsymbol{h}^{\text{Cross}} = \mathbb{FC}(f^{\text{Cross}}(\boldsymbol{e}^{\text{Cross}})) \tag{7}$$

where $\mathbb{FC}$ denotes one MLP layer for dimension reduction. f^{Cross} denotes a mixture of low-rank cross network blocks. For a detailed

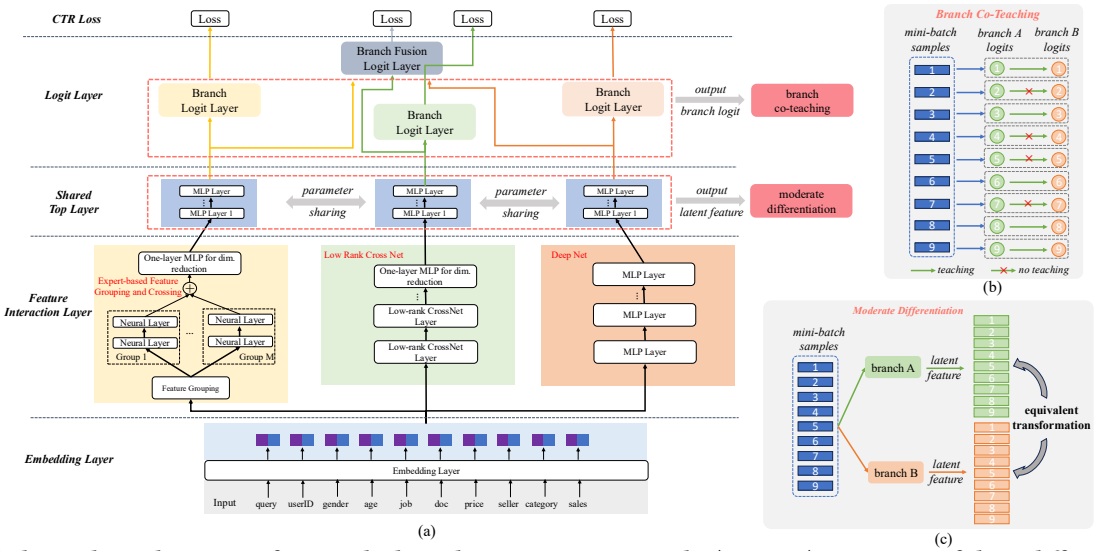


Figure 1: (a) shows the architecture of our multi-branch cooperation networks (MBCnet). It consists of three different branches: the EFGC branch, the low-rank Cross Net branch and the Deep net branch. (b) illustrates the branch co-teaching between any two of the three branches. (c) demonstrates that latent features from any two branches undergo equivalent transformation.

formulation, please refer to [29]. After the feature interaction learning of each branch, we further include a shared top layer $f_\theta(\cdot)$ to reduce parameters and regularize training. The output latent features of different branches are formulated as:

$$\boldsymbol{z}^{\text{EFGC}} = f_\theta(\boldsymbol{h}^{\text{EFGC}}),\ \boldsymbol{z}^{\text{Deep}} = f_\theta(\boldsymbol{h}^{\text{Deep}}),\ \boldsymbol{z}^{\text{Cross}} = f_\theta(\boldsymbol{h}^{\text{Cross}}) \tag{8}$$

where $\boldsymbol{z}^{\text{EFGC}}$, $\boldsymbol{z}^{\text{Deep}}$ and $\boldsymbol{z}^{\text{Cross}}$ are also the input of logit layer.

3.4 Multi-branch Cooperation

3.4.1 ***Branch Co-teaching***. Different branches possess distinct advantages and inherently focus on modeling different patterns. These patterns, in turn, are represented by the training samples. As a result, the learning capabilities of different branches can vary significantly across different samples. On particular samples, if a robust branch has been effectively trained, it can assist a comparatively weak branch in enhancing its performance. We formulate this collaboration idea as "***branch co-teaching***" objective function.

Sample Selection by Disagreement: To perform the above co-teaching, we first need to identify on which samples that one branch exhibits strong prediction performance while the other one shows weak performance [33]. In other words, two branches have disagreement predictions on these samples. In CTR prediction, it is naturally to use binary cross entropy (BCE) loss to measure whether a model has accurate prediction on one sample. To be specific, we denote the predicted click probability of one branch as:

$$\boldsymbol{p}^{\boldsymbol{i}} = \sigma(f_\phi^i(\boldsymbol{z}^{\boldsymbol{i}})) \tag{9}$$

where $i \in \{\text{EFGC, Deep, Cross}\}$, $\sigma(\cdot)$ denotes the sigmoid function. $f_\phi^i(\cdot)$ and $\boldsymbol{z}^{\boldsymbol{i}}$ denote the logit layer and latent feature of branch i, respectively. Therefore on one particular sample, we can identify strong and weak branches as:

$$(\boldsymbol{p}^{\boldsymbol{j}} \text{ is strong}, \boldsymbol{p}^{\boldsymbol{i}} \text{ is weak})\ \textit{if}\ \begin{cases} BCE(\boldsymbol{p}^{\boldsymbol{j}}, \boldsymbol{y}) < -\log(0.5) \\ BCE(\boldsymbol{p}^{\boldsymbol{i}}, \boldsymbol{y}) > -\log(0.5) \end{cases} \tag{10}$$

$$(\boldsymbol{p}^{\boldsymbol{i}} \text{ is strong}, \boldsymbol{p}^{\boldsymbol{j}} \text{ is weak})\ \textit{if}\ \begin{cases} BCE(\boldsymbol{p}^{\boldsymbol{i}}, \boldsymbol{y}) < -\log(0.5) \\ BCE(\boldsymbol{p}^{\boldsymbol{j}}, \boldsymbol{y}) > -\log(0.5) \end{cases} \tag{11}$$

where *BCE* means binary cross entropy loss. Since CTR prediction is usually regarded as a binary classification task, we employ $-\log(0.5)$ as a threshold to assess the learning progress of each branch[1]. When the loss value is less than $-\log(0.5)$, we consider the branch on this samples is learned effectively. Otherwise, we consider the branch is not learned well.

Branch Co-teaching Objective Formulation: Given the above defined strong and weak branches, we take the prediction of strong branches as soft label to guide the learning of weak branches on selected samples. The co-teaching loss is written as follows:

$$\mathcal{L}_{BCT} = -\frac{1}{C}\sum_i \sum_{j \neq i} [I^{ij}\text{SG}(\boldsymbol{p}^{\boldsymbol{i}})\log(\boldsymbol{p}^{\boldsymbol{j}}) + I^{ji}\text{SG}(\boldsymbol{p}^{\boldsymbol{j}})\log(\boldsymbol{p}^{\boldsymbol{i}})] \tag{12}$$

where $i, j \in \{\text{EFGC, Deep, Cross}\}$. SG means stop gradient. $I^{ij} \in \{0, 1\}$ is the indicator that only equals 1 when $\boldsymbol{p}^{\boldsymbol{i}}$ is strong and $\boldsymbol{p}^{\boldsymbol{j}}$ is weak shown in Eq. 11. Similarly, $I^{ji} \in \{0, 1\}$ is the indicator that equals 1 only when $\boldsymbol{p}^{\boldsymbol{j}}$ is strong and $\boldsymbol{p}^{\boldsymbol{i}}$ is weak shown in Eq. 10. In other cases, both I^{ij} and I^{ji} equal 0, which means we will not perform co-teaching optimization on those samples. *This suggests that when two learners do not produce significantly different predictions, transferring teaching guidance may be unnecessary. It allows both learners the flexibility to discover and capture their own patterns.* We conducted an experiment to demonstrate this idea in Section 4.4. C is the number of non-zero values calculated from I^{ij} and I^{ji}, indicating how many disagreement pairs are used in $\mathcal{L}_{BCT}$.

3.4.2 ***Moderate Differentiation***. When multiple branches are supervised on the same sample, they have risk to learn identical feature interaction patterns. This redundancy can significantly hinder the model's capacity to explore a variety of feature interactions.

[1]The prediction probabilities may be affected by the data distribution, we experimented with various thresholds and found $-\log(0.5)$ generally did the best.

To enhance the model's robustness, it is better to enable distinct branches to learn differentiated latent features, encouraging them to capture diverse interaction patterns even on the same input. *However, excessive differentiation between these latent features may disrupt the consistency of shared patterns across branches, potentially harming model performance.* In this context, we propose "***moderate differentiation***" for the latent features z^i across branches. This approach promotes the exploration of diverse interaction patterns while ensuring a balance between differentiation and consistency.

Equivalent Transformation of Latent Features: To achieve moderate differentiation between any two branches, we assume that their latent features satisfy equivalent transformation [8]. This indicates these two latent features have relations but also learning flexibility to ensure differentiation [3]. Specifically, if $z^i \in \mathbb{R}^{1\times d}$ and $z^j \in \mathbb{R}^{1\times d}$ are the latent features from two distinct branches on the same sample, they satisfy the equivalent transformation as[2]:

$$z^i = z^j W^{ji},\ z^j = z^i W^{ij},\ s.t.\ W^{ji}W^{ij} = I_d \tag{13}$$

where $W^{ji}, W^{ij} \in \mathbb{R}^{d\times d}$ are learnable matrices and they are inverses of each other. To avoid feature collapse where multiple z^i collapse to one point after transformation, we employ orthogonal transformation, a specification of equivalent transformation. This means W^{ji} and W^{ij} are the transpose to each other, *i.e.*, $W^{ij} = (W^{ji})^T$.

Moderate Differentiation Regularization: Taking the above analysis into consideration, we can write the regularization loss for moderate differentiation as:

$$\mathcal{L}_{MDR} = \frac{1}{K(K-1)} \sum_i \sum_{j\neq i} ||z^i W^{ij} - z^j||_F^2 + ||z^i W^{ij}(W^{ij})^T - z^i||_F^2 \tag{14}$$

where the first term indicates the transformation relationship between z^i and z^j, while the second term shows the constraint of orthogonal transformation on W^{ij}. *This loss not only enhances the model's capability but also facilitates the learning process by maintaining relevant feature representations across branches.*

3.4.3 ***Branch Fusion and Training Objective Function***. We perform branch fusion to incorporate their knowledge and make better behavior predictions. Specifically, given latent feature of three branches, we have:

$$z^{\text{fusion}} = Avg\_Pool(z^{\text{EFGC}}, z^{\text{Deep}}, z^{\text{Cross}}) \tag{15}$$

where z^{fusion} is the fused latent feature of different branches. *Avg_Pool* means average pooling of latent features in feature dimension. Then, the training objective for CTR prediction is formulated as:

$$\mathcal{L}_{CTR} = \sum_i BCE(\sigma(f_\phi^i(z^i)), \boldsymbol{y}) \tag{16}$$

where $i \in \{\text{fusion, EFGC, Deep, Cross}\}$ and *BCE* is the binary cross entropy loss. $f_\phi^i(\cdot)$ indicates the logit layer. $\sigma(\cdot)$ denotes the sigmoid function. It is also worthwhile to mention that BCE loss is one choice for CTR prediction, it can be replaced with other CTR loss functions. To sum up, the overall objective function of MBCnet is defined as:

$$\mathcal{L} = \mathcal{L}_{CTR} + \alpha * \mathcal{L}_{BCT} + \beta * \mathcal{L}_{MDR} \tag{17}$$

where α and β are two hyper-parameters to weight the importance of loss terms. Notice that we use an arbitrary sample to succinctly demonstrate above objective function with simple symbols. When using the objective function in practice, we compute the expectation of loss values of mini-batch samples during training.

[2]Note that we use an simplified version of equivalent transformation as in [3]

Table 2: The statistics of used industrial datasets.

Dataset	I2P-12month	I2P-24month
#user	480,028,940	528,665,908
#item	974,333,635	1,355,865,204
#feature_fields	192	192
#feature_dim	2780	2780
#samples	203.99 billion	395.99 billion
density	4.36e-5%	5.52e-4%

3.5 Model Time Complexity Analysis

We have made a complexity analysis of different branches/components in MBCnet. The time complexity of EFGC branch is $O(N_g L_{\text{EFGC}} d^2)$, where N_g is the number of feature groups, L_{EFGC} is the number of EFGC layers and d denotes the latent dimension. The time complexity of CrossNet branch is $O(L_{\text{CrossNet}} N_{\text{experts}} Fd)$, where L_{CrossNet} is the number of CrossNet layers, N_{experts} denotes the number of experts and F is the input feature dimension. For Deep branch, the time complexity is $O(L_{\text{Deep}} d^2)$, where L_{Deep} is the number of mlp layers. Considering the branch fusion in Eq.15 is average pooling and the complexity is negligible, so the complexity of MBCnet can be taken as the summation of three branches. Fortunately, different components of MBCnet works in a parallel style so that we can use parallel computation to accelerate training and inference. In our online deployment, compared to the base model(DCNv2), the real-time latency only has 1ms increase and the GPU utility only has 2% increase, which are negligible in practice.

4 Experiment and Analysis

4.1 Experiment Setup

4.1.1 Datasets. We conduct experiments on two industrial datasets: I2P-12month and I2P-24month. Both are collected from our image2product search of our e-commerce app. I2P-12month and I2P-24month contain user search behaviors ranging from 2023-06-01 to 2024-05-31 and 2022-06-01 to 2024-05-31, respectively. For both datasets, there are various feature fields, including query image feature, user profiles, user statistical features, item image feature, item attributes, item statistical features and some other designed clustering and crossing features. The dataset statistics are given in the supplementary. All experimental models utilize the same input features to ensure a fair comparison of their modeling differences. The dataset statistics are shown in Table 2.

4.1.2 Baselines. We leverage different strong algorithms for comparison including **single branch methods** and **branch ensemble** methods. The **single branch methods** are as follows. 1) **DNN** is a deep multi-layer perception model, 2) **CrossNetV2** [29] is a low-rank version of cross net [27]. The **branch ensemble methods** are as follows. 3) **Wide&Deep** [4] combines a wide set of cross-product feature transformations and deep neural networks. 4) **DCNv2** [29] contains mixture-of-expert based low-rank feature crossing and DNN. 5) **MMOE+** [16] originally is a multi-task learning model. We employ its mixture-of-expert encoder to work as a baseline with

Table 3: The AUC and LogLoss comparison results of different models on our two datasets. Best results are in bold and the most competitive public baseline results are underlined.

Dataset		I2P-12month		I2P-24month	
Model		AUC↑	LogLoss↓	AUC↑	LogLoss↓
Single	DNN	0.7423	0.2894	<u>0.7569</u>	<u>0.2829</u>
	CrossNetV2	0.7437	0.2917	0.7555	0.2835
Ensemble	Wide&Deep	0.7392	0.2911	0.7513	0.2863
	DCNv2	<u>0.7461</u>	0.2872	0.7559	0.2837
	MMOE+	0.7400	0.2906	0.7544	0.2837
	EnKD	0.7450	<u>0.2871</u>	0.7541	0.2834
	FinalMLP	0.7427	0.2891	0.7556	0.2838
Ours	EFGC	0.7490	0.2880	0.7600	0.2825
	MBCnet	**0.7522 (+0.61%)**	**0.2866 (-0.05%)**	**0.7642 (+0.72%)**	**0.2800 (-0.29%)**

Table 4: The comparison results when deploying MBCnet in our production environment. AUC is the offline evaluation metric. CTR, deal number and GMV are online A/B test metrics. Some values of the base model are denoted as ⋆ to obey the company's regulations.

Model	AUC	CTR	deal number by query	deal number by user	GMV
Base	0.7559	10.37%	⋆	⋆	⋆
MBCnet	**0.7642**	**10.46%(+0.09point)**	**+1.49%**	**+1.18%**	**+1.62%**

multiple-expert ensemble. 6) **EnKD** [36] studies knowledge distillation technique for model ensemble. 7) **FinalMLP** [17] designs feature gating and interaction aggregation for CTR prediction.

4.1.3 Parameter Settings. We split the datasets by chronological order, with the last seven-day data as validation and test set, and the rest is taken as the train set. For all methods, we set the latent dimension of last feature learning layer as 128. The validation performance is set as early-stop condition in training. In MBCnet, α and β both equal 0.1 by hyper-parameter searching. In EFGC branch, hidden units of each group crossing are [1024,128]. In CrossNet branch, the number of experts is 2 and each expert has 2 layers with low rank dimension as 16. The hidden units of dimension reduction layer in EFGC and CrossNet are both 512. In Deep branch, the hidden units are [2048,1024,512,512,512]. Hidden units of the shared top layer are [512,256,128].

4.2 Overall Comparison

4.2.1 ***Offline Comparison.*** We make model comparisons with recent competitive baselines on our two large-scale industrial datasets. The results are given in Table 3. From this table, we can observe that: 1) our MBCnet achieves obvious improvement over recent competitive baselines. It has a 0.61% and 0.72% AUC increase over the most competitive model on two datasets, which are impressive improvements in large-scale industrial data scenarios. 2) Even when solely using our EFGC model, it can achieve superior model performance over other baselines, demonstrating its effectiveness.

4.2.2 ***Online Production Deployment.*** We further evaluate the method through online A/B testing in image2product search of our e-commerce system. In this experiment, the training data is our I2P-24month dataset. The baseline is our previously online serving model (DCNv2). Online evaluation metrics are real CTR, deal number by query, deal number by user and GMV. Table 4 shows that our MBCnet has an absolute **0.09 point** CTR increase, a relative **1.49%** deal growth and a relative **1.62%** GMV increase. *Since August 2024, MBCnet has been fully deployed online, serving hundreds of millions of customers.*

Table 5: Study of the cooperation principles to demonstrate how to do multi-branch cooperation.

Study Principle	Dataset	I2P-12month		I2P-24month	
	Model Variant	AUC	LogLoss	AUC	LogLoss
Principle 1	no discrimination	0.5772	0.3251	0.5575	0.3319
	weak to strong	0.4933	0.3254	0.4754	0.3617
	strong to weak	**0.7522**	**0.2866**	**0.7642**	**0.2800**
Principle 2	max difference	0.7176	0.3498	0.7366	0.3360
	min difference	0.7285	0.3447	0.7349	0.3138
	moderate differentiation	**0.7522**	**0.2866**	**0.7642**	**0.2800**

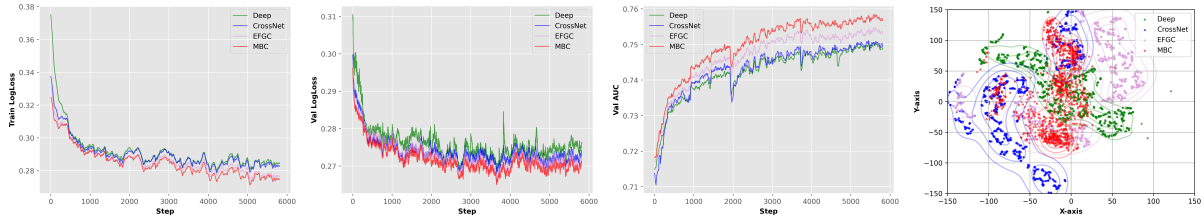
4.3 Why Using Multi-Branch Cooperation

In this part, we conduct an experiment to demonstrate why using multi-branch cooperation in our industrial business. In particular, we show some learning dynamics of different branches during training. Moreover, we compare the feature distribution before logit layer across branches with t-SNE [26]. The results are summarized in Figure 2, where "EFGC", "Deep", "CrossNet" are three branches and "MBC" is the fused one.

From Figure 2 (a)-(c), we see that: 1) The fused "MBC" model consistently achieves lower train log loss and validation log loss, along with a higher validation AUC throughout the training process. It indicates that a single branch has limited learning capacity, whereas MBCnet enhances its learning ability by cooperating multiple branches. 2) As shown in Figure 2 (d), the output features from "EFGC," "Deep," and "CrossNet" branches are distinct from one another. In contrast, the fused "MBC" features are more evenly distributed across the feature space. We have also observed the output features across branches are more entangled and hard to be identified in early training stage, while the features tend to cluster to different subspaces and the fused "MBC" is becoming more evenly distributed in later training epochs. **These cooperative dynamics demonstrate that each branch has its own modeling patterns, and MBCnet can gradually integrate their diverse capabilities. More experiments about why using multi-branch cooperation are given in Sec 4.5.5.**

4.4 How to do Multi-Branch Cooperation

We also explore various approaches in multi-branch cooperation to validate the effectiveness of our proposed two principles: ***branch co-teaching*** and ***moderate differentiation***. To examine the first principle, we modify our "strong to weak" learning scheme described in Eq. 10-12 to create two variants. The first variant, labeled "no discrimination", does not perform sample selection during co-teaching; instead, each branch can teach the other using the entire dataset. The second variant, labeled "weak to strong", reverses the teaching direction by having the weak branch instruct the strong branch. To investigate the second principle, we adjust the formulation of "moderate differentiation" from Eq. 14 in two ways. The first one, called "max difference", aims to maximize the L2 distance between latent features of branches. The second one, referred to as



(a) Train LogLoss (b) Val LogLoss (c) Val AUC (d) Feature Distribution

Figure 2: Comparison results of different branches on our I2P-12month dataset. (a)-(c) show the performance comparison of different branches in training. (d) indicates the out feature distribution of different branches after model convergence.

Table 6: Ablation study of different model components.

Dataset	I2P-12month		I2P-24month	
Model	AUC	LogLoss	AUC	LogLoss
w/o EFGC	0.7445	0.2888	0.7548	0.2839
w/o CrossNet	0.7480	0.2880	0.7569	0.2828
w/o DeepNet	0.7501	0.2875	0.7622	0.2811
w/o $\mathcal{L}_{\text{BCT}}$	0.7510	0.2874	0.7628	0.2806
w/o $\mathcal{L}_{\text{MDR}}$	0.7506	0.2879	0.7603	0.2822
w/o $\mathcal{L}_{\text{BCT}}$, w/o $\mathcal{L}_{\text{MDR}}$	0.7443	0.2954	0.7609	0.2818
MBCnet	**0.7522**	**0.2866**	**0.7642**	**0.2800**

"min difference", seeks to minimize the L2 distance between branch features. The results are summarized in Table 5.

In Table 5, we show that violating our proposed scheme leads to a decline in model performance. Principle 1 is particularly crucial because it dictates the selection of knowledge to be transferred among branches. Principle 2 also affects performance by ensuring branch diversity and encouraging the model to discover different patterns.

4.5 Ablation Study

4.5.1 Study of Different Model Parts. To evaluate the impact of each model component, we performed an ablation study by systematically removing specific modules, as detailed in Table 6. *The results indicate a clear decline in performance on both datasets when any component is removed.* For instance, excluding the EFGC branch leads to a 0.94% decrease in AUC on I2P-24month. This result highlights the importance of our EFGC branch. The mechanism in EFGC highlights feature crossing within feature groups, which encourages the model to capture specific patterns and meanwhile reduces redundant crossing features. Additionally, removing the cooperation scheme, *i.e.*, "w/o $\mathcal{L}_{\text{BCT}}$, w/o $\mathcal{L}_{\text{MDR}}$", results in a 0.79% reduction in AUC on I2P-12month. These outcomes demonstrate that each component positively contributes to the overall performance.

4.5.2 The Convergence of MBCnet. We also show some key learning metrics during training to show the convergence and rationality of our MBCnet in Figure 3. In Figure 3, it reveals that both the co-teaching loss and the moderate differentiation loss converge, while the validation AUC steadily increases. In the converged state shown in Figure 3 (c), the difference between equivalent transformation matrix W and I_d remains non-zero, demonstrating that the branches maintain distinct discrepancies to explore diverse patterns.

Table 7: The AUC comparison of different models under different data density on I2P-12month. The listed density ratio is relative to the original train set.

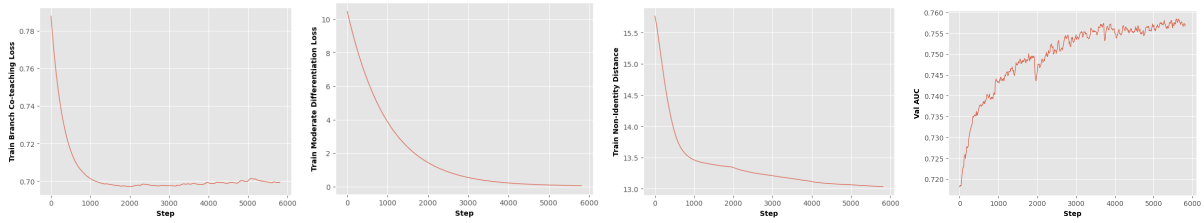
Density ratio	20%	40%	60%	80%	100%
DNN	0.5132	0.5885	0.6770	0.7223	0.7423
DCNv2	0.5200	0.5929	0.6797	0.7262	0.7461
MBCnet	**0.5277**	**0.6015**	**0.6890**	**0.7319**	**0.7522**

4.5.3 Hyper-Parameter Sensitivity. In MBCnet, the hyper-parameters α and β regulate the weighting of the loss functions $\mathcal{L}_{\text{BCT}}$ and $\mathcal{L}_{\text{MDR}}$, respectively. To understand the impact of these parameters, we performed a sensitivity analysis, and the results are illustrated in Figure 4. From the figure, we observe that: 1) Comparing the results in Figure 4 with those in Table 3, it is clear that our model can perform better than the baselines across a wide range of α and β values. This robustness highlights the effectiveness of incorporating cooperative interactions between different branches of the model. 2) The performance of the model degrades if α and β are set either too high or too low. Specifically, excessively large values for these hyper-parameters can dominate the primary CTR task, while too small values may make the cooperative effects between branches negligible.

4.5.4 Performance Under Different Data Sparsity. We also conducted performance comparison under different data density. Specifically, we keep the validation and test set fixed, and mask a certain ratio of positive samples. The AUC results of some methods on I2P-12month are given in Table 7. The result shows that MBCnet consistently achieves better performance even in more sparse cases.

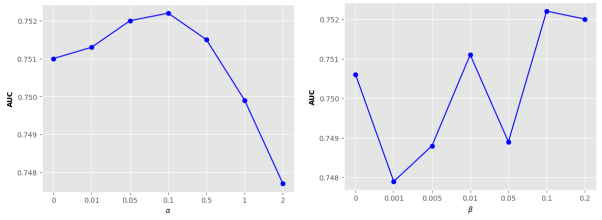
4.5.5 Branch Difference in Learning. **In the Neuron Level**: We design an experiment to illustrate the branch difference at neuron level. In particular, MBCnet integrates latent features from multiple branches to execute the final CTR prediction task. These latent features are derived from the outputs of neurons, as depicted in Figure 5 (a). Each dimension of a feature value reflects the strength of a specific high-order feature. To analyze these feature strengths for each branch, we normalize the neuron outputs using softmax function[3]. The normalized distributions are shown in Figure 5 (b)[4].

[3]Our network has an output dimension of 128. For clarity, we randomly select 30 neuron outputs to illustrate the results.
[4]Notice that the output features from different branches are in the same space in our network design, enabling direct comparison of neuron values across branches.



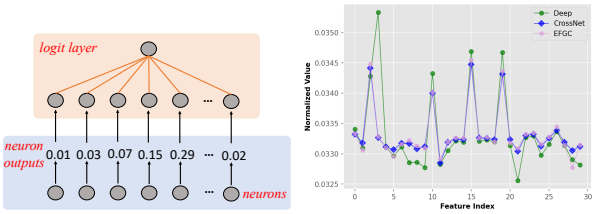
(a) Train Co-Teaching Loss (b) Train Differentiation Loss (c) Difference between W and I_d (d) Val AUC

Figure 3: Different train and val metrics along training on our I2P-12month dataset. (a) and (b) show the train branch co-teaching loss $\mathcal{L}_{BCT}$ and train moderate differentiation loss $\mathcal{L}_{MDR}$, respectively. (c) indicates the Euclidean distance between equivalent transformation matrix W and the identity matrix I_d. (d) shows the val auc of our MBCnet in different training steps.



(a) (b)

Figure 4: Hyper-parameter sensitivity on I2P-12month.



(a) (b)

Figure 5: (a) is an example to illustrate the structure of neuron outputs before logit layer. (b) shows the output distribution of those neurons on I2P-12month.

From the figure, we have the following observations: 1) Different branches have close outputs at most neurons while quite different values at few neurons. This indicates different branches have many common learned patterns and several specific patterns, which aligns with our moderate differentiation principle. *The common patterns ensure branches are learned towards the same learning objective, while the specific patterns encourage the model to develop complementary functionalities, benefiting the final prediction task.* 2) We also observe that "EFGC" branch exhibits greater consistency with "CrossNet" branch compared to "Deep" branch. One possible reason for this may be that the "EFGC" branch engages in feature crossing by domain-driven knowledge, which can be regarded as one kind of explicit feature crossing in "CrossNet".

In the Category Level: In this part, we conduct an experiment to highlight the branch learning difference at category level. Specifically, for each branch, we first sort the samples in ascending order based on their logloss values, where a lower log loss value indicates an accurately learned sample. We then select the top 10,000 samples and count them under each category to get the category distribution of these accurately learned samples for each branch. The results are presented in Figure 6.

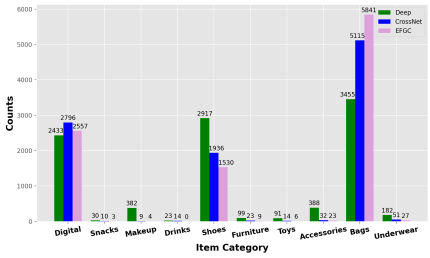


Figure 6: The category distribution of top-10k most accurately learned samples (*i.e.*, those with low log loss) across different branches in our MBCnet.

From this figure, we clearly see that *different branches have unique learning strengths and prefer distinct categories.* For example, "Deep" branch outperforms both "CrossNet" and "EFGC" branches in learning "Makeup", "Shoes", "Accessories" and "Underwear" categories, but it exhibits weaker performance in "Digital" and "Bags" categories. In contrast, the proposed "EFGC" branch excels over the other two branches in "Bags" category but falls behind in "Shoes" and "Snacks" categories. *These findings highlight the unique advantages of each branch and their complementary roles across different types of data.*

5 Conclusion and Future Work

In this paper, we introduced Multi-Branch Cooperation Network (MBCnet) for enhanced CTR prediction. MBCnet effectively integrates three distinct branches (*i.e.*, EFGC, low rank CrossNet and Deep Net) in capturing complex feature interactions. *The proposed cooperation scheme, based on the principles of* ***branch co-teaching*** *and* ***moderate differentiation****, facilitates meaningful collaboration between branches. It also enables the model to explore diverse feature interaction patterns and improve the overall prediction accuracy.* Our extensive experiments, including an online A/B test, demonstrated significant improvements in CTR, deal growth, and GMV, validating its effectiveness and scalability in real-world applications.

Currently we simply use the common average pooling and use later MLP layer to implicitly combine the outputs of branches. We believe that a better way to fuse the branch outputs could lead to better performance. Meanwhile, we use loss values to classify branches as strong or weak on particular samples. This may produce unreliable loss measurements during the early training stage, potentially limiting the model's learning ability. We will study these issues in future work.

References

[1] Xu Chen, Siheng Chen, Jiangchao Yao, Huangjie Zheng, Ya Zhang, and Ivor W. Tsang. 2022. Learning on Attribute-Missing Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 2 (2022), 740–757. doi:10.1109/TPAMI.2020.3032189

[2] Xu Chen, Zida Cheng, Jiangchao Yao, Chen Ju, Weilin Huang, Jinsong Lan, Xiaoyi Zeng, and Shuai Xiao. 2024. Enhancing Cross-Domain Click-Through Rate Prediction via Explicit Feature Augmentation. In *Companion Proceedings of the ACM on Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 423–432. doi:10.1145/3589335.3648341

[3] Xu Chen, Ya Zhang, Ivor W Tsang, Yuangang Pan, and Jingchao Su. 2020. Towards equivalent transformation of user preferences in cross domain recommendation. *ACM Transactions on Information Systems (TOIS)* (2020).

[4] 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*. 7–10.

[5] Binzong Geng, Zhaoxin Huan, Xiaolu Zhang, Yong He, Liang Zhang, Fajie Yuan, Jun Zhou, and Linjian Mo. 2024. Breaking the length barrier: Llm-enhanced CTR prediction in long textual user behaviors. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2311–2315.

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence* (Melbourne, Australia) *(IJCAI'17)*. AAAI Press, 1725–1731.

[7] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising* (New York, NY, USA) *(ADKDD'14)*. Association for Computing Machinery, New York, NY, USA, 1–9. doi:10.1145/2648584.2648589

[8] Jim Hefferon. 2018. Linear algebra third edition. (2018).

[9] Zhaoxin Huan, Ke Ding, Ang Li, Xiaolu Zhang, Xu Min, Yong He, Liang Zhang, Jun Zhou, Linjian Mo, Jinjie Gu, et al. 2024. Exploring Multi-Scenario Multi-Modal CTR Prediction with a Large Scale Dataset. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1232–1241.

[10] Rohit Kumar, Sneha Manjunath Naik, Vani D Naik, Smita Shiralli, Sunil V.G, and Moula Husain. 2015. Predicting clicks: CTR estimation of advertisements using Logistic Regression classifier. In *2015 IEEE International Advance Computing Conference (IACC)*. 1134–1138. doi:10.1109/IADCC.2015.7154880

[11] Pan Li and Alexander Tuzhilin. 2020. DDTCDR: Deep Dual Transfer Cross Domain Recommendation. *International conference on web search and data mining* (2020).

[12] Xiangyang Li, Bo Chen, Lu Hou, and Ruiming Tang. 2023. Ctrl: Connect collaborative and language model for ctr prediction. *ACM Transactions on Recommender Systems* (2023).

[13] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1754–1763. doi:10.1145/3219819.3220023

[14] Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. ClickPrompt: CTR Models are Strong Prompt Generators for Adapting Language Models to CTR Prediction. In *Proceedings of the ACM on Web Conference 2024*. 3319–3330.

[15] 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* (Perth, Australia) *(WWW '17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. doi:10.1145/3041021.3054192

[16] 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*. 1930–1939.

[17] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 4 (Jun. 2023), 4552–4560. doi:10.1609/aaai.v37i4.25577

[18] Ibomoiye Domor Mienye and Yanxia Sun. 2022. A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. *IEEE Access* 10 (2022), 99129–99149. doi:10.1109/ACCESS.2022.3207287

[19] Aashiq Muhamed, Iman Keivanloo, Sujan Perera, James Mracek, Yi Xu, Qingjun Cui, Santosh Rajagopalan, Belinda Zeng, and Trishul Chilimbi. 2021. CTR-BERT: Cost-effective knowledge distillation for billion-parameter teacher models. In *NeurIPS Efficient Natural Language and Speech Processing Workshop*.

[20] Wentao Ouyang, Xiuwu Zhang, Li Li, Heng Zou, Xin Xing, Zhaojie Liu, and Yanlong Du. 2019. Deep spatio-temporal neural networks for click-through rate prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2078–2086.

[21] Wentao Ouyang, Xiuwu Zhang, Lei Zhao, Jinmei Luo, Yu Zhang, Heng Zou, Zhaojie Liu, and Yanlong Du. 2020. Minet: Mixed interest network for cross-domain click-through rate prediction. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 2669–2676.

[22] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. 995–1000. doi:10.1109/ICDM.2010.127

[23] 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* (Beijing, China) *(CIKM '19)*. Association for Computing Machinery, New York, NY, USA, 1161–1170. doi:10.1145/3357384.3357925

[24] Jingchao Su, Xu Chen, Ya Zhang, Siheng Chen, Dan Lv, and Chenyang Li. 2020. Collaborative Adversarial Learning for Relational Learning on Multiple Bipartite Graphs. In *2020 IEEE International Conference on Knowledge Graph (ICKG)*. 466–473. doi:10.1109/ICBK50248.2020.00072

[25] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems*. 269–278.

[26] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9, 86 (2008), 2579–2605. http://jmlr.org/papers/v9/vandermaaten08a.html

[27] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17* (Halifax, NS, Canada) *(ADKDD'17)*. Association for Computing Machinery, New York, NY, USA, Article 12, 7 pages. doi:10.1145/3124749.3124754

[28] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.

[29] 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 2021*. 1785–1797.

[30] Xinfei Wang. 2020. A Survey of Online Advertising Click-Through Rate Prediction Models. In *2020 IEEE International Conference on Information Technology,Big Data and Artificial Intelligence (ICIBA)*, Vol. 1. 516–521. doi:10.1109/ICIBA50161.2020.9277337

[31] Yongquan Yang, Haijun Lv, and Ning Chen. 2022. A Survey on ensemble learning under the era of deep learning. *Artificial Intelligence Review* 56, 6 (Nov. 2022), 5545–5589. doi:10.1007/s10462-022-10283-5

[32] Yanwu Yang and Panyu Zhai. 2022. Click-through rate prediction in online advertising: A literature review. *Information Processing & Management* 59, 2 (2022), 102853. doi:10.1016/j.ipm.2021.102853

[33] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. 2019. How does disagreement help generalization against label corruption?. In *International conference on machine learning*. PMLR, 7164–7173.

[34] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

[35] 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*. 1059–1068.

[36] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Virtual Event, Ireland) *(CIKM '20)*. Association for Computing Machinery, New York, NY, USA, 2941–2958. doi:10.1145/3340531.3412704

Listing 1: Loss Code of MBCnet in Tensorflow Style

```
import tensorflow as tf
# binary cross entropy loss
def bce_loss_op(self, logit, label, reduction='mean'):
    bce_loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=logit,labels=label))
    return bce_loss
# branch co-teaching loss
def BCT_loss_op(self, logits_list, labels):
    losses_list = []
    for logits in logits_list:
        loss = self.bce_loss_op(logits, labels, reduction='none')
        losses_list.append(loss)
    BCT_loss, n_sample = 0.0, 1e-8
    for i in range(len(logits_list)):
        for j in range(i,len(logits_list)):
            if i!=j:
                # mask of disagreement data
                mask_i = tf.logical_and(tf.reshape(tf.cast(losses_list[i] < -tf.log(0.5), tf.bool), [-1]), tf.
                    reshape(tf.cast(losses_list[j] > -tf.log(0.5), tf.bool),[-1]))
                mask_j = tf.logical_and(tf.reshape(tf.cast(losses_list[j] < -tf.log(0.5), tf.bool), [-1]),tf.
                    reshape(tf.cast(losses_list[i] > -tf.log(0.5), tf.bool),[-1]))
                # label supervision from strong branch
                BCT_loss += tf.cond(tf.reduce_sum(tf.cast(mask_j,tf.float32))>0, lambda:self.bce_loss_op(tf.
                    boolean_mask(logits_list[i],mask_j),tf.stop_gradient(tf.boolean_mask(tf.sigmoid(logits_list[j
                    ]),mask_j)),reduction='sum'), lambda:tf.zeros_like(self.reg_loss))
                BCT_loss += tf.cond(tf.reduce_sum(tf.cast(mask_i,tf.float32))>0, lambda:self.bce_loss_op(tf.
                    boolean_mask(logits_list[j],mask_i),tf.stop_gradient(tf.boolean_mask(tf.sigmoid(logits_list[i
                    ]),mask_i)),reduction='sum'), lambda:tf.zeros_like(self.reg_loss))
                n_sample += tf.reduce_sum(tf.cast(mask_j,tf.float32)) + tf.reduce_sum(tf.cast(mask_i,tf.float32))
    BCT_loss = BCT_loss/n_sample
    return BCT_loss
# moderate differentiation loss
def MDR_loss_op(self, feats_list):
    mdr_loss_list, hidden_dim = [], feats_list[0].get_shape().as_list()[1]
    for i in range(len(feats)):
        for j in range(len(feats)):
            if i!=j:
                feat_i, feat_j = feats[i], feats[j]
                # orthogonal transformation constraint, W_{ji} is the transpose of W_{ij}
                mapped_feat_i_to_j = tf.matmul(feat_i, self.params['W_{}{}'.format(i,j)])
                mapped_feat_i_to_j_to_i = tf.matmul(mapped_feat_i_to_j, self.params['W_{}{}'.format(j,i)])
                l2_distance = tf.norm(mapped_feat_i_to_j-feat_j, ord='euclidean', axis=1) + tf.norm(
                    mapped_feat_i_to_j_to_i-feat_i, ord='euclidean', axis=1)
                mdr_loss_list.append(l2_distance)
    mdr_loss = tf.reduce_mean(tf.concat(mdr_loss_list, 0))
    return mdr_loss
# The overall loss op
def loss_op(self, branch_logits_list, branch_top_feats_list, fused_logits, labels):
    # branch_logits_list: branch output logits, [efgc, crossnetv2, dnn], each one is [B, 1]
    # branch_top_feats_list: branch output features, [efgc, crossnetv2, dnn], each one is [B, d]
    # fused_logits: the fused branch logits, [B, 1]; labels: the labels of samples, [B, 1]
    efgc_logits, crossnetv2_logits, dnn_logits = branch_logits_list
    with tf.name_scope("CTR_Loss_Op"):
      efgc_bce_loss = self.bce_loss_op(efgc_logits) # EFGC bce loss
      crossnetv2_bce_loss = self.bce_loss_op(crossnetv2_logits) # CrossNetV2 bce loss
      dnn_bce_loss = self.bce_loss_op(dnn_logits) # DNN bce loss
      fused_bce_loss = self.bce_loss_op(fused_logits) # fused bce loss
      self.bce_loss = efgc_bce_loss + crossnetv2_bce_loss + dnn_bce_loss + fused_bce_loss # overall bce loss
      # multi-branch cooperation loss
      self.BCT_loss, self.MDR_loss = self.BCT_loss_op(branch_logits_list, labels), self.MDR_loss_op(
          branch_top_feats_list)
      self.loss = self.bce_loss + self.alpha * self.BCT_loss + self.beta * self.MDR_loss # final loss
```