Fusion Matters: Learning Fusion in Deep Click-through Rate Prediction Models

Kexin Zhang*
Northwestern University
Evanston, IL, USA
kevin.kxzhang@gmail.com

Fuyuan Lyu*
McGill University & MILA
Montreal, Canada
fuyuan.lyu@mail.mcgill.ca

Xing Tang†
FiT, Tencent
Shenzhen, China
xing.tang@hotmail.com

Dugang Liu†
Shenzhen University
Shenzhen, China
dugang.ldg@gmail.com

Chen Ma
City University of Hong Kong
Hong Kong SAR, China
chenma@cityu.edu.hk

Kaize Ding
Northwestern University
Evanston, IL, USA
kaize.ding@northwestern.edu

Xiuqiang He
FiT, Tencent
Shenzhen, China
xiuqianghe@tencent.com

Xue Liu
McGill University
Montreal, Canada
xueliu@cs.mcgill.ca

Abstract

The evolution of previous Click-Through Rate (CTR) models has mainly been driven by proposing complex components, whether shallow or deep, that are adept at modeling feature interactions. However, there has been less focus on improving fusion design. Instead, two naive solutions, stacked and parallel fusion, are commonly used. Both solutions rely on pre-determined fusion connections and fixed fusion operations. It has been repetitively observed that changes in fusion design may result in different performances, highlighting the critical role that fusion plays in CTR models. While there have been attempts to refine these basic fusion strategies, these efforts have often been constrained to specific settings or dependent on specific components. Neural architecture search has also been introduced to partially deal with fusion design, but it comes with limitations. The complexity of the search space can lead to inefficient and ineffective results. To bridge this gap, we introduce OptFusion, a method that automates the learning of fusion, encompassing both the connection learning and the operation selection. We have proposed a one-shot learning algorithm tackling these tasks concurrently. Our experiments are conducted over three large-scale datasets. Extensive experiments prove both the effectiveness and efficiency of OptFusion in improving CTR model performance. Our code implementation is available here[1].

CCS Concepts

• **Information systems** → **Social recommendation**; **Computational advertising**.

Keywords

CTR Prediction; Fusion Learning; Connection Learning; Neural Architecture Search

ACM Reference Format:
Kexin Zhang, Fuyuan Lyu, Xing Tang, Dugang Liu, Chen Ma, Kaize Ding, Xiuqiang He, and Xue Liu. 2025. Fusion Matters: Learning Fusion in Deep Click-through Rate Prediction Models. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25), March 10–14, 2025, Hannover, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3701551.3703557

1 Introduction

Click-through rate (CTR) prediction is a vital task for commercial recommender systems and online advertising platforms, as it seeks to predict the likelihood that a user will click on a recommended item, such as a movie or advertisement [2, 29]. As deep learning-based CTR models have advanced, researchers have developed various model architectures [5, 14, 26, 36, 37, 43] to better capture feature interactions and enhance prediction performance. These deep CTR models employ a combination of explicit and implicit components to represent feature interactions. Shallow components, including inner products [26], cross layer [36], and Factorization Machines (FM) [5, 28], are used to model these interactions explicitly. Concurrently, deep components like multi-layer perceptrons (MLP) [43] and Self-attention layer [32] implicitly capture the complexity of feature interaction. With the improved deep and shallow components, deep CTR models can model feature interactions more effectively, leading to better prediction accuracy.

Despite advances in CTR prediction through the enhancement of deep and shallow components, the design of fusion mechanisms has not been extensively studied. Fusion design is crucial as it involves

*Both authors contributed equally to this research. Work done when they were research interns at FiT, Tencent.
†Corresponding Authors
[1]https://github.com/kexin-kxzhang/OptFusion

WSDM '25, March 10–14, 2025, Hannover, Germany

ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703557

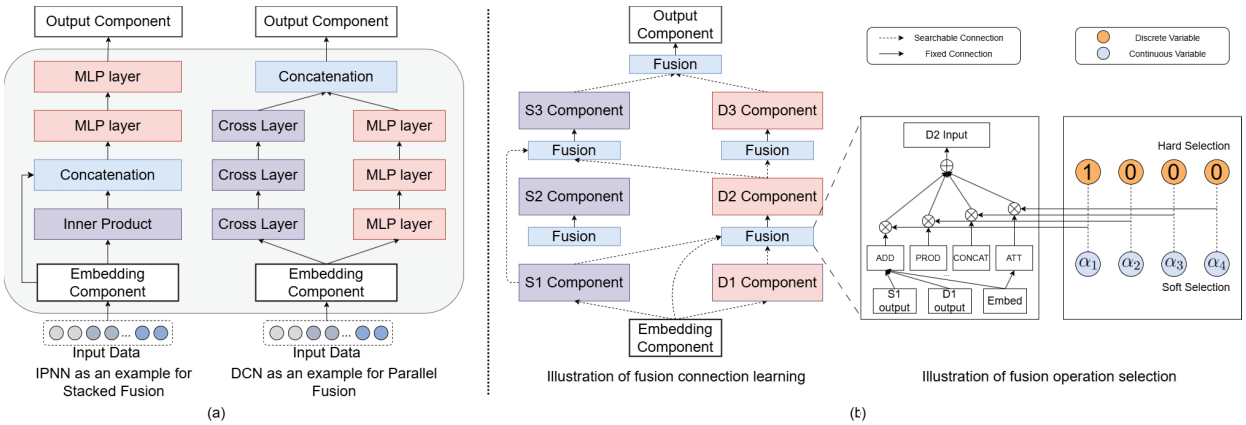


Figure 1: (a) Illustration figure about two mainstream fusion designs in deep CTR models. (b) The framework of OptFusion, which consists of one embedding component, n cross components, n deep components, and one output component as the candidate set of fusion connection search. The selection of components is formed as a Directed Acyclic Graph (DAG). Fusion operation selection is also designed to fuse representations from lower-level components.

aggregating representations from different model components. Previous works [26, 36], as illustrated in Figure 1, have predominantly relied on two naive fusion designs: *stacked* and *parallel*. In the stacked design, shallow components are typically placed before deep components and are trained sequentially [7, 15, 26]. For example, as depicted in Figure 1 (a), IPNN [26] uses the inner product as a shallow component, concatenating its output with original embeddings before feeding them into deep components. Conversely, the parallel design involves the joint training of shallow and deep components. Models like DCN [36] use concatenation for fusion, while others, such as DeepFM [5] and Wide&Deep [4], employ addition to combine outputs from both components [4, 14, 32]. In summary, these two naive fusion designs rely on **pre-defined fusion connections** and **fixed fusion operations** to fuse representations from both deep and shallow components. However, variations in fusion design can lead to substantial differences in performance across different datasets. For instance, DCNv2 [37] with a parallel design may outperform its stacked counterpart on the MovieLens dataset while underperforming on the Criteo dataset. This inconsistent performance among different fusion designs across various datasets is also observed in MaskNet [38]. These findings underscore the critical role of fusion in CTR predictions.

Attempts have been made to refine the above-mentioned fusion designs. EDCN [3], for instance, introduces a manually crafted complex fusion design that incorporates additional fusion connections and sophisticated fusion operations. Despite using the same components as its predecessor DCN [36], EDCN achieves significant improvement in performance. FinalMLP [24] suggests the use of Multi-Head Bilinear Fusion Operations as a more effective means of integrating representations, outperforming naive fusion operations such as concatenation or addition. These examples demonstrate the potential of fusion design in significant improvements. However, these proposed solutions [3, 24] typically address the fusion learning problem under specific settings or depending on specific modules, failing to demonstrate the potential of fusion learning explicitly. A more general and adaptable approach to fusion learning remains a compelling challenge.

Other researchers have explored using neural architecture search (NAS) within CTR models to deal with fusion design challenges by defining a broader search space [31, 42]. AutoCTR [31] employs an evolutionary approach to simultaneously search for the optimal component type, raw feature input, fusion connections, and component-specific hyperparameters. However, this approach incurs a considerable training cost since each candidate architecture must be trained separately during the search phase. NASRec [42] takes a different approach by expanding the search space even further for better utility while leveraging weight-sharing techniques to mitigate the training cost during the search process. Despite these innovations, searching for fusion design in conjunction with other model structures, such as component types, within such a vast search space can make it much harder to obtain an optimal result. In contrast, our research focuses solely on one of the critical aspects: fusion learning. By narrowing down the search space, better results and faster convergence speed can be achieved. The issues mentioned above highlight the necessity for a comprehensive yet lightweight approach to fusion learning that simultaneously selects both fusion connections and operations.

To address these challenges, we introduce **OptFusion**, an automated fusion learning framework for deep CTR prediction. OptFusion aims to explore how fusion, both in terms of connections and operations, can impact CTR predictions and automatically identify the most suitable fusion design. We propose a unified search space specifically tailored to the fusion learning process with shallow and deep components. This design allows each component to be connected to its predecessors. Such a design effectively focuses the search space on fusion learning and facilitates more efficient exploration compared to neural architecture search methods in CTR models. Drawing inspiration from previous work [30], we propose a one-shot learning algorithm that concurrently learns fusion connections and selects fusion operations. This algorithm considers

the entanglement and mutual influence between fusion connections and operations, leading to better selection results. Through rigorous evaluation on three widely-used benchmarks, we empirically demonstrate the effectiveness and efficiency of OptFusion. Additionally, our ablation studies have shown the orthogonality of OptFusion when combined with various components. We summarize our contributions as follows:

- We investigate the importance of fusion, both connections and operations. A novel CTR fusion learning framework, namely OptFusion, is proposed. OptFusion can automatically learn suitable fusions in CTR models, containing both connection learning and operation selection.
- To better explore the proposed fusion space, a one-shot learning algorithm that simultaneously selects fusion connection and operation is proposed. Such a one-shot learning algorithm can better explore the search space due to entanglement between fusion operation and connection.
- We empirically evaluate OptFusion's efficiency and effectiveness on three large-scale datasets.

2 Preliminary

In this section, we first formulate the CTR prediction problem in Section 2.1. Then, we introduce the fusion learning for CTR prediction in Section 2.2.

2.1 CTR Prediction

CTR prediction is a classic supervised binary classification problem [29]. Given a dataset $\mathcal{D} = \{(\mathbf{x}, y)\}$ consisting of $N = |\mathcal{D}|$ instances with each containing a pair of user and item, the CTR prediction aims to predict whether the user would click the item. Here $\mathbf{x}$ denotes the input data instance, and $y \in \{0, 1\}$ denotes the label indicating whether the user clicked the item.

In deep learning-based CTR prediction models, an embedding layer $\mathcal{E}$ is usually adopted to transform the input $\mathbf{x}$ with high-dimensional sparse raw features into low-dimensional dense embeddings $\mathbf{e}$. The embedding $\mathbf{e}$ can be obtained via embedding lookup [5], formulated as $\mathbf{e} = \mathcal{E}(\mathbf{x})$. The embedding $\mathbf{e}$ is further fed into the CTR model, formulated as follows:

$$\hat{y} = \mathcal{F}\left(\mathbf{e}, \Theta\right) = \mathcal{F}\left(\mathcal{E}(\mathbf{x}), \Theta\right) \tag{1}$$

where $\hat{y}$ is the probability that a user will click a given item, $\mathcal{F}(\cdot)$ is the prediction model, and Θ is the corresponding trainable model parameters. Cross-entropy loss is commonly adopted to train Θ, which can be formulated as follows:

$$\arg\min_{\Theta} \mathcal{L}_{\mathcal{D}}(\Theta) = \sum_{(\mathbf{x},y)\in\mathcal{D}} y \log \hat{y} + (1-y)\log\left(1-\hat{y}\right). \tag{2}$$

2.2 Fusion Learning in CTR

In this section, we aim to take a deeper look at the CTR model $\mathcal{F}(\cdot)$. Here, we formulate the CTR model as an instance of the fusion design, written as:

$$\mathcal{F} = \mathcal{P}(\mathcal{G}|\mathbf{c}, \mathbf{o}). \tag{3}$$

Here $\mathcal{P}$ denotes the fusion learning, parameterized by fusion connection $\mathbf{c}$ and fusion operation $\mathbf{o}$. $\mathcal{G} = \{\mathcal{G}^s, \mathcal{G}^d\}$ refer to the set of all components in the CTR model, with $\mathcal{G}^s$ and $\mathcal{G}^d$ represent the set of shallow and deep components, respectively. Various shallow components such as cross layer [36], factorization machine [5], or inner product [26] can be chosen. Similarly, deep components may also vary from MLP layer [43] to self-attention module [32]. Below, we separately discuss the two parameters of fusion learning: fusion connection $\mathbf{c}$ and fusion operation $\mathbf{o}$.

2.2.1 Fusion Connection. Fusion connection $\mathbf{c}$ determines the connectivity between different components. Such connectivity determines what information is fed into the current component. There are only two potential states of connectivity between two components, i.e., `CONNECTED` or `DISCONNECTED`. Hence, we adopt a connectivity function $\mathbf{c}(\cdot)$ to formulate the fusion connection from component $\mathcal{G}_i$ to component $\mathcal{G}_j$ as

$$\mathbf{c}\left(\mathcal{G}_i, \mathcal{G}_j\right) = \begin{cases} 1, & \text{if } \texttt{CONNECTED} \\ 0, & \text{if } \texttt{DISCONNECTED} \end{cases} \tag{4}$$

In the fusion learning, each component has its level, which constrains the direction of the connection. The current component only takes the outputs from lower-level components as the input for the fusion module. Such design is introduced to avoid cycles [30, 31]. Such a constraint can be formulated on each $\left(\mathcal{G}_i, \mathcal{G}_j\right)$ pairs:

$$\mathbf{c}\left(\mathcal{G}_i, \mathcal{G}_j\right) = 1 \rightarrow L(i) < L(j), \forall \left(\mathcal{G}_i, \mathcal{G}_j\right) \tag{5}$$

where $L(i)$ is a non-negative integer denotes the level of component $\mathcal{G}_i$. It increases monotonically as the component gets deeper.

2.2.2 Fusion Operation. After determining the fusion connection, a fusion operation o needs to be selected for each component. The fusion operation aggregates the output representations from the connected components. It outputs a fused representation, which serves as the input for the succeeding component $\mathcal{G}_j$, as shown in Figure 1. This process can be written as:

$$\hat{\mathbf{e}}_j = o_j(\{\mathbf{c}\left(\mathcal{G}_i, \mathcal{G}_j\right) \cdot \mathbf{e}_i\}). \tag{6}$$

Here $\hat{\mathbf{e}}_j$ refers to the input for component $\mathcal{G}_j$ and $\mathbf{e}_i$ denotes the output for preceding component $\mathcal{G}_i$. Note that the fusion operations are usually selected from a set of candidates O. It may vary from simple operations such as `ADD` or `CONCAT` to complex operations like Multi-Head Bilinear Fusion Ops [24]. Given the one-to-one mapping relationship between fusion operation o_j and component $\mathcal{G}_j$, we have $|\mathbf{o}| = |\mathcal{G}|$.

2.2.3 Fusion Learning in CTR. After formulating fusion connection learning and fusion operation selection, the prediction in Eq. 1 can be reformulated as:

$$\hat{y} = \mathcal{F}(\mathcal{E}(x), \Theta) = \mathcal{P}(\mathcal{G}|\mathbf{c}, \mathbf{o})(\mathcal{E}(x), \Theta). \tag{7}$$

Consequently, the training objective in Eq. 2 can be rewritten as:

$$\arg\min_{\Theta,\ \mathbf{c}\in\{0,1\},\ \mathbf{o}\in O} \mathcal{L}_{\mathcal{D}}(\Theta, \mathbf{c}, \mathbf{o}) \tag{8}$$

3 OptFusion

In this section, we detail the OptFusion framework under the fusion learning setting. We first describe the search space of the framework in Section 3.1. Then, we introduce fusion connection learning in Section 3.2 and fusion operation selection in Section 3.3. Finally, we elaborate on the details of the one-shot learning algorithm, which jointly conducts fusion connection learning and fusion operation selection for OptFusion, in Section 3.4.

3.1 Search Space

In this section, we detail the search space of the framework in the fusion learning setting. The OptFusion framework consists of one embedding component $\mathcal{E}(\cdot)$, n shallow components, n deep components, and one output component $\mathcal{H}(\cdot)$. The number of all components in the search space is $2n+2$. The default configuration of the OptFusion framework with $n=3$ is illustrated in Figure 1 (b). The setting of n is discussed in Section 4.4.4.

Candidates of Fusion operation. Shallow and deep components may receive multiple inputs from lower-level components. Thus, a fusion operation is needed to fuse these inputs. The commonly-used fusion operations include ADD, PROD, CONCAT and ATT, which represent element-wise addition, Hadamard product, concatenation, and attention, respectively. Details are shown in Table 1. Each component can select one of them to fuse information from lower-level components. Alternatively, each component can also output a weighted sum. Depending on this, we propose two variants of OptFusion, namely *Hard* and *Soft*.

Table 1: A summary of different fusion operations.

Operation	Description	Formulation
ADD	Element-wise addition	$\mathbf{x}' = \mathbf{x}_i + \mathbf{x}_j$
PROD	Hadamard product	$\mathbf{x}' = \mathbf{x}_i \odot \mathbf{x}_j$
CONCAT	Concatenation	$\mathbf{x}' = \mathbf{W}[\mathbf{x}_i \| \mathbf{x}_j]$
ATT	Attention mechanism	$\mathbf{x}' = a_i \cdot \mathbf{x}_i + a_j \cdot \mathbf{x}_j$, $a_i = \frac{\exp(\mathbf{w}_2^T \mathrm{ReLU}(\mathbf{W}_1 \mathbf{x}_i + \mathbf{b}_1))}{\sum_j \exp(\mathbf{w}_2^T \mathrm{ReLU}(\mathbf{W}_1 \mathbf{x}_j + \mathbf{b}_1))}$

Take an example to illustrate the fusion operation, two vectors $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathbf{x}_j \in \mathbb{R}^d$ are used as inputs, and $\mathbf{x}' \in \mathbb{R}^d$ is the output of fusion. In PROD, $\odot$ denotes the Hadamard product operation. In CONCAT, $||$ denotes the concatenation operation, and $\mathbf{W} \in \mathbb{R}^{d \times 2d}$ is the trainable weight parameter. In ATT, a_i and a_j are attention coefficients, $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$, $\mathbf{w}_2 \in \mathbb{R}^d$ and $\mathbf{b}_1 \in \mathbb{R}^d$ are the trainable weight and bias parameters.

Search space analysis. In this subsection, we intuitively illustrate the difficulty in selecting suitable fusions. Given that OptFusion aims to search for both fusion connections and operations, we need to jointly consider their possibility. The number of possible connections is determined by the number of components $2n+2$. The number of all valid connections equals to $2 \times (1+3+\cdots+2n-1) + 2(n+1) = 2n^2+2n+1$. For each valid connection, the number of choices for its connection state is 2. Thus, the size of the search space for connections is 2^{2n^2+2n+1}. For each component, suppose the number of possible choices for its fusion operation is k. Thus, the search space size for the fusion operation is k^{2n+1}. The number of possible fusions equals $2^{2n^2+2n+1} \times k^{2n+1} = \mathbf{O}(2^{2n} \times k^n)$. Directly selecting over such a large space is almost impossible. Hence, we separately discuss the connection learning and operation selection in the following two sections.

3.2 Fusion Connection Learning

A critical issue for fusion connection learning lies in the discrete selection space. Searching within a discrete candidate set of connections (i.e., $C = \{\texttt{CONNECTED}, \texttt{DISCONNECTED}\}$) is non-differentiable, which makes the architecture untrainable. To solve this problem, we relax the discrete search space to be continuous by learning the relative importance (i.e., probability) of each connection and introduce the architecture parameters $\boldsymbol{\alpha} \in \mathbb{R}^{(2n+2)^2}$ to parameterize the connectivity function $\mathbf{c}(\cdot)$ so that the fusion connection becomes learnable. With $\boldsymbol{\alpha}_{ij}$ representing the connectivity $\mathbf{c}(\mathcal{G}_i, \mathcal{G}_j)$ from component $\mathcal{G}_i$ to $\mathcal{G}_j$, Eq. 4 can be rewritten as follows

$$\boldsymbol{\alpha}_{ij} \begin{cases} > 0, & \text{if CONNECTED} \\ \leq 0, & \text{if DISCONNECTED} \end{cases}, \ 1 \leq i, j \leq 2n+2. \tag{9}$$

To satisfy the level constraint in Eq. 5, $\boldsymbol{\alpha}$ is constrained as:

$$\boldsymbol{\alpha}_{ij} \geq 0 \rightarrow L(i) < L(j), \ \forall 1 \leq i, j \leq 2n+2. \tag{10}$$

To enable end-to-end training and get meaningful gradients for $\boldsymbol{\alpha}$, we adopt the straight-through estimator (STE) function [1]. The STE can be formulated as a customized function S(·), with its forward pass as a unit step function $S(x) = 0, x \leq 0$ and $S(x) = 1, x > 0$. $S(x)$'s backward pass equals to $\frac{d}{dx}\mathcal{S}(x) = 1$, meaning that it will directly pass the gradient backward. Therefore, we can mimic a discrete selection while providing valid gradient information for connection parameters $\boldsymbol{\alpha}$, making the whole process trainable. Hence, the final output in Eq. 7 can be rewritten as:

$$\hat{y} = \mathcal{P}(\mathcal{G}|\boldsymbol{\alpha}, \mathbf{o})(\mathcal{E}(x), \Theta). \tag{11}$$

3.3 Fusion Operation Selection

For shallow or deep components, they may receive multiple connections from lower-level components. One operation needs to be selected from the set of fusion operations to fuse the information received from lower-level components. Specifically, we define the operation candidates as $O = \{\texttt{ADD}, \texttt{PROD}, \texttt{CONCAT}, \texttt{ATT}\}$ and $|O| = k$. Similar to connection learning, we also relax the discrete search space of fusion operations to be continuous by learning the relative importance of each operation and introduce the architecture parameters $\boldsymbol{\beta} \in \mathbb{R}^{k \times (2n+2)}$ to represent the operation selection.

Given a component j, we assign an architecture parameter β_j^o to an operation $o \in O$, the importance of operation o is computed as a *softmax* of all candidate operations $o' \in O$:

$$p_j^o = \exp\left(\beta_j^o\right) / \sum_{o' \in O} \exp\left(\beta_j^{o'}\right), \tag{12}$$

where p_j^o is the importance of operation o. During the selection stage, the input of component j equals a weighted summation over all candidate operations:

$$\hat{\mathbf{e}}_j = \sum_{o \in O} p_j^o \cdot o\left(\{\boldsymbol{\alpha}_{ij} \cdot \mathbf{e}_i\}\right), \tag{13}$$

where $\mathbf{e}_i$ is the output of component i. $o(\cdot)$ fuses all the inputs by using operation o. Finally, by parameterizing the selection of fusion operation $o \in O$ with architecture parameters $\boldsymbol{\beta}$, Eq. 11 can be rewritten as follows:

$$\hat{y} = \mathcal{P}(\mathcal{G}|\boldsymbol{\alpha}, \boldsymbol{\beta})(\mathcal{E}(x), \Theta). \tag{14}$$

3.4 One-shot Learning Algorithm

After obtaining Eq. 14, which parameterizes the fusion learning via architecture parameter $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, we need to rewrite the original learning goal in Eq. 8 to incorporate the fusion learning process. The optimization process can be rewritten as:

$$\min_{\Theta, \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}} \mathcal{L}_{\mathcal{D}}\left(\Theta, \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}\right) \tag{15}$$

We can observe that the parameters that need to be optimized include the following three categories:

- Θ, the model parameters, including the parameters in all components and the transformation matrices between components.
- $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, the architecture parameters of fusion design, representation connection learning and operation selection, respectively.

Firstly, to disentangle the model and architecture parameters, we follow the paradigm from NAS [18, 31] by first learning the architecture parameters $\{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$ and conduct retraining to get model parameter Θ given the architecture parameters.

Secondly, the entanglement between fusion connection and fusion operation remains a challenge. An intuitive approach is determining fusion connection $\boldsymbol{\alpha}$ and fusion operation $\boldsymbol{\beta}$ sequentially. However, such a design omits the mutual influence between connection and operation, as selecting inappropriate operations would decrease the likelihood of connections.

Finally, we formulate the one-shot learning algorithm, which jointly and simultaneously conducts connection learning and operation selections given their entanglement. The one-shot learning algorithm consists of two stages: *selection stage* and *re-train stage*. In the selection stage, connection learning and operation selection are achieved by simultaneously optimizing the architecture parameters of connections $\boldsymbol{\alpha}$ and fusion operations $\boldsymbol{\beta}$. In the re-train stage, the architecture parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are fixed, and the model parameter Θ is re-trainted.

3.4.1 Selection Stage. The goal in the selection stage is to jointly learn $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ given their mutual information. This allows the model to explore different connections and fusion operations during the selection process. The optimization can be formulated as below:

$$\boldsymbol{\alpha}^*, \boldsymbol{\beta}^* = \arg\min_{\Theta, \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}} \mathcal{L}_{\mathcal{D}} \left(\Theta, \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}\right) \tag{16}$$

3.4.2 Re-train Stage. In the retraining stage, the parameters need to be optimized only to include the model parameters Θ. We keep the selected architecture parameters $\boldsymbol{\alpha}^*$ and $\boldsymbol{\beta}^*$ fixed, re-train the model parameters Θ to obtain the final model. Following previous works [18], the selected connection tensor is determined as $\alpha^* = \mathbb{1}_{\alpha_{i,k}>0}$ during the re-training stage. With the different ways to conduct fusion operations based on the score, we propose two variants of the proposed model, i.e., OptFusion-Soft and OptFusion-Hard, which refer to soft selection and hard selection of the fusion operations, respectively.

OptFusion-Soft. In OptFusion-Soft, fusion operations are performed in a soft manner, i.e., combining the weighted summation over all candidate operations according to Equation 13. The final architecture parameter for fusion can be formulated as $\beta^* = \beta$.

OptFusion-Hard. The final fusion operation type is selected with the largest weight based on the learned fusion parameters. This is formalized as: $\beta_k^{o*} = 1$ if $o = \arg\max_{o \in O} \beta_k^o$ and $\beta_k^{o*} = 0$ otherwise.

After obtaining the architecture parameters α^* and β^* for fusion connection and operation. The model parameters are then re-trained with fixed architecture parameters.

$$\Theta^* = \arg\min_{\Theta} \mathcal{L}_{\mathcal{D}} \left(\Theta, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*\right) \tag{17}$$

This one-shot selection algorithm allows OptFusion to efficiently explore different architectures during the selection stage and then fine-tune the discovered architecture in the re-train stage. Finally, the pseudo-code of the learning algorithm for OptFusion is summarized in Algorithm 1.

Algorithm 1 The OptFusion Algorithm

Require: Training dataset $\mathcal{D}$ consisting original features x and ground-truth labels y
Ensure: Learned architecture parameter $\boldsymbol{\alpha}^*$, $\boldsymbol{\beta}^*$, model parameters Θ^*

```
Selection Stage
t=0
while t < T do
    t = t + 1
    while not converged do
        Sample a mini-batch B from the training dataset D
        Update the model parameters Θ, connection parameters α and operation parameters β by Eq. 16
    end while
end while
Re-train Stage
Retrain Θ given α*, β* by Eq. 17
```

4 Experiments

In this section, to comprehensively validate OptFusion, we design and conduct various experiments over three large-scale datasets, aiming to answer the following research questions:

- **RQ1**: Could OptFusion achieve superior performance compared with mainstream deep CTR prediction models?
- **RQ2**: How efficient is OptFusion compared with mainstream deep CTR prediction models?
- **RQ3**: How does the selection of fusion operation influence the performance?
- **RQ4**: How effective is the one-shot selection algorithm?
- **RQ5**: How compatible is OptFusion with existing components?
- **RQ6**: How does the number of components affect performance?
- **RQ7**: Does OptFusion select the suitable fusion?

4.1 Experimental Setting

4.1.1 Datasets. To demonstrate the effectiveness of OptFusion, we evaluate our model on three real-world datasets, the statistics of datasets are described in Table 2.

Table 2: Statistics of datasets.

Dataset	#samples	#Fields	#Values	pos ratio
Criteo	4.6×10^7	39	6.8×10^6	0.2562
Avazu	4.0×10^7	24	4.4×10^6	0.1698
KDD12	1.5×10^8	11	6.0×10^6	0.0445

Note: *#samples* refers to the total samples in the dataset, *#field* refers to the number of feature fields for original features, *#value* refers to the number of feature values for original features, *pos ratio* refers to the positive ratio.

Criteo[2] dataset consists of ad click data over a week. It consists of 26 categorical feature fields and 13 numerical feature fields. Following the winner solution of the Criteo Advertising Challenge [9], we discretize each numeric value x to $\lfloor \log^2(x) \rfloor$, if $x > 2$; $x = 1$

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

otherwise. We replace infrequent categorical features with a default "OOV" (i.e. out-of-vocabulary) token, with *min_count*=2.

Avazu[3] dataset contains 10 days of click logs. It has 24 fields with categorical features. We remove the *instance_id* field and transform the *timestamp* field into three new fields: *hour*, *weekday* and *is_weekend*. We replace infrequent categorical features with the "OOV" token, with *min_count*=2.

KDD12[4] dataset contains training instances derived from search session logs. It has 11 categorical fields, and the click field is the number of times the user clicks the ad. We replace infrequent features with an "OOV" token, with *min_count*=2.

4.1.2 Metrics. To evaluate the performance of CTR Prediction, we adopt the most commonly-used evaluation metrics [5], i.e., **AUC** (Area Under ROC) and **LogLoss** (cross-entropy). Note that **0.1**% improvement in AUC is considered significant [5, 26, 41].

4.1.3 Baselines. To demonstrate the effectiveness of OptFusion, we compare the performance with four categories of deep CTR prediction models, including (i) stacked models: FNN [43] and PNN [26], DCNv2s [37]; (ii) parallel models: DeepFM [5], DCN [36], xDeepFM [14], DCNv2p [37][5]; (iii) models with expert design on fusion: EDCN [3]; (iv) NAS models: AutoCTR [31], NASRec [42].

4.1.4 Implementation Details. We use Adam [12] as the optimizer for all models and set the embedding size as 40 for the Criteo and Avazu datasets and 16 for the KDD12 dataset. The batch size is fixed at 4096. Following [3], we employ a three-layer MLP with the number of neurons equal to $dim_{emb} \times num_{field}$. We also incorporate an auxiliary shallow block $S0$ within the candidate set of connections to mimic the stacked structure. We search the optimal learning rate from {3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5} and L_2 regularization from {3e-6, 1e-6, 3e-7, 1e-9, 0}. For OptFusion, during the re-training phase, we reuse the optimal learning rate and L_2 regularization obtained in the initial training. α in connection search is initialized as 0.5 to ensure equal weight for each block at the start. Hyperparameters used in the experiments are reported in [6], and the implementation of our algorithm is available here[7]. For NAS models, we reuse the official implementation[8] with the same embedding size and batch size as ours.

4.2 Overall Performance (RQ1)

The overall performance of our OptFusion and other deep CTR prediction models on three datasets are reported in Table 3. We summarize the observations as follows:

First, OptFusion, both soft and hard, outperforms all the SOTA baselines over three datasets in terms of both AUC and Logloss by a significant margin. This demonstrates that OptFusion can effectively find a suitable fusion connection and operation. It also echoes our intuition: fusion learning is an important but overlooked aspect of feature interaction modeling. Specifically, OptFusion improves AUC over the best baseline by 0.0011, 0.0021, and 0.0036 on three datasets, respectively.

Table 3: The overall performance comparison.

Method	Criteo		Avazu		KDD12	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
FNN	0.8037	0.4473	0.7860	0.3766	0.7978	0.1534
PNN	0.8048	0.4463	0.7886	0.3752	0.8011	0.1527
DCNv2s	0.8088	0.4427	0.7877	0.3753	0.8030	0.1536
DeepFM	0.8038	0.4486	0.7856	0.3806	0.7963	0.1532
DCN	0.8063	0.4450	0.7875	0.3779	0.7968	0.1537
xDeepFM	0.8067	0.4453	0.7860	0.3773	0.7966	0.1542
DCNv2p	0.8085	0.4451	0.7894	0.3759	0.8012	0.1531
EDCN	0.8102	0.4419	0.7917	0.3727	0.8122	0.1498
AutoCTR	0.8082	0.4436	0.7883	0.3761	0.7949	0.1533
NASRec	0.8090	0.4435	0.7893	0.3752	0.7958	0.1530
OptFu.-H	0.8108*	0.4413*	0.7935*	0.3717*	0.8129	0.1496
OptFu.-S	**0.8113***	**0.4408***	**0.7938***	**0.3715***	**0.8158***	**0.1489***
Impr	0.0011	0.0023	0.0021	0.0012	0.0036	0.0009

Here $*$ denotes statistically significant improvement (measured by a two-sided t-test with p-value < 0.05) over the best baseline. Bold scores are the best performance, and underlined scores are the best baseline performance. OptFu.-H and OptFu.-S stand for OptFusion-Hard and OptFusion-Soft, respectively.

Second, OptFusion, with a smaller search space, outperforms NASRec and AutoCTR, which contain a larger search space. Such an observation demonstrated that OptFusion could better exploit and explore the fusion search space, while NASRec and AutoCTR's huge search space could potentially lead to sub-optimal results.

Third, EDCN, which aims to fuse explicit and implicit information densely, constantly performs as the best baseline over all three datasets. This proves that the naive fusion design is an obstacle towards accurate prediction, echoing previous observations [3].

Finally, the performance of naive fusions varies across datasets. For example, on the Avazu datasets, DCNv2p, a parallel model, exhibits superior performance. On the Criteo and KDD12 datasets, DCNv2s, a stack model, outperforms other baselines. This interesting observation further reveals the limitation of naive fusion design, which we will discuss in Section 4.5.

4.3 Efficiency Analysis (RQ2)

In addition to model effectiveness, training, and inference efficiency are crucial considerations when deploying CTR prediction models in practice. In this section, we investigate the time complexity of OptFusion. Due to the expansive search space of AutoCTR and NASRec, training efficiency experiments are conducted on an NVIDIA A40 GPU with 48G memory, while inference efficiency experiments are conducted on an NVIDIA RTX 4090 GPU with 24G memory.

We illustrate the total training time of NAS models trained on all three datasets in Figure 2 (a). Here, the total training time encompasses both the search and re-train stages. We observe that OptFusion achieves the shortest total training time compared to other NAS models. This is attributed to the narrowed search space for OptFusion and the adoption of a one-shot learning algorithm.

As depicted in Figure 2 (b-d), we plot the Inference Time-AUC curve of mainstream deep CTR models trained on three datasets, indicating the relationship between time complexity and model performance. Compared with models such as DCN, DeepFM, and PNN, which achieve the least inference time, both EDCN and OptFusion take fusion design into consideration, and they tend to achieve

[3] http://www.kaggle.com/c/avazu-ctr-prediction
[4] http://www.kddcup2012.org/c/kddcup2012-track2/data
[5] DCNv2s refers to a stacked design while DCNv2p involves a parallel design in the original paper [37].
[6] https://github.com/kexin-kxzhang/OptFusion/hyperparam.md
[7] https://github.com/kexin-kxzhang/OptFusion
[8] https://github.com/facebookresearch/nasrec

the highest AUC. This can be attributed to the trade-off between inference time and model performance.

4.4 Ablation Study

4.4.1 Fusion Operation (RQ3). In this section, we aim to investigate how the fusion operation influences the performance of deep CTR models. We reuse the searched fusion connection and replace the searched fusion operation with four identical fusion operations: ADD, PROD, CONCAT, and ATT, as introduced in Table 1. These models corresponding to four fusion operations are retrained from scratch. Results of OptFusion-soft and OptFusion-hard selection, which adopt different operations for different blocks, in Table 3 are also listed for easy comparison. The results are shown in Table 4.

We can easily observe that both Soft and Hard methods exhibit significantly superior performance compared to models with fixed fusion operations. This verifies the effectiveness of our fusion operation search instead of using a fixed fusion operation.

In addition, ADD and PROD operations outperform the others. This may be attributed to the fact that element-wise addition and Hadamard products are parameter-free operations, making them easier to train steadily compared to methods incorporating parameters like concatenation and attention pooling.

Table 4: Ablation study on fusion operation.

Operation	Criteo		Avazu		KDD12	
	AUC	Logloss	AUC	Logloss	AUC	Logloss
Add	0.8111	0.4422	0.7872	0.3970	0.7924	0.1585
Product	0.8077	0.4443	0.7860	0.3784	0.7938	0.1584
Concatenate	0.8075	0.4445	0.7837	0.3814	0.7926	0.1546
Attention	0.8073	0.4442	0.7843	0.3794	0.7883	0.1597
Hard	0.8108	0.4413	0.7935	0.3717	0.8129	0.1496
Soft	**0.8113**	**0.4408**	**0.7938**	**0.3715**	**0.8158**	**0.1489**

4.4.2 Selection Algorithm (RQ4). In this section, we investigate the search algorithm design. We aim to compare the one-shot selection algorithm, which jointly selects both the connection and operation simultaneously, with a sequential selection algorithm, which sequentially selects the connection and operation. Experiments are conducted over Criteo and Avazu datasets, and the results are shown in Table 5.

Table 5: Ablation study on selection algorithm.

Methods	Criteo		Avazu	
	AUC	Logloss	AUC	Logloss
One-shot	**0.8113**	**0.4408**	**0.7938**	**0.3715**
Sequential	0.8109	0.4411	0.7934	0.3717

Results indicate that our one-shot selection algorithm performs better than the sequential selection algorithm. This gap is likely caused by the mutual influence between fusion connection and operation. Such an observation verifies the effectiveness of OptFusion and the one-shot selection algorithm.

4.4.3 Shallow Component (RQ5). In this section, we conduct an ablation study on the compatibility of OptFusion over various explicit components. In the default setting, we adopt CrossNet [36] as the explicit component for OptFusion. We further replace CrossNet with CrossNetV2 [37] and CIN [14], and construct its two variants: OptFusion-CrossNetV2 and OptFusion-CIN. The results are summarized in Table 6. We additionally adopt the fusion connection and operation from EDCN as a comparison for all three explicit components, namely EDCN, EDCN-CrossNetV2, and EDCN-CIN.

Table 6: Ablation study on shallow components.

Method	Criteo		Avazu	
	AUC	Logloss	AUC	Logloss
DCN	0.8063	0.4450	0.7895	0.3762
EDCN	0.8102	0.4419	0.7917	0.3727
OptFu.-Soft	**0.8113**	**0.4408**	**0.7938**	**0.3715**
DCNv2	0.8085	0.4451	0.7903	0.3751
EDCN-CrossNetV2	0.8091	0.4434	0.7923	0.3723
OptFu.-CrossNetV2	**0.8111**	**0.4408**	**0.7942**	**0.3717**
xDeepFM	0.8067	0.4453	0.7860	0.3773
EDCN-CIN	0.8089	0.4426	0.7943	0.3712
OptFu.-CIN	**0.8112**	**0.4408**	**0.7955**	**0.3703**

OptFu. is an abbreviation for OptFusion.

From the table, we can easily observe that OptFusion and its two variants achieve the best performance on both datasets. These results underscore the robustness and compatibility of OptFusion to different explicit components. Besides, EDCNs constantly rank the 2nd, outperforming the original models. This further indicates the importance of dense fusion in deep CTR models.

4.4.4 Number of Components (RQ6). This section evaluates the impact of varying the number of components (n) on OptFusion's performance. The default setting for n is 3. To investigate the effect of different configurations, we also conduct experiments with $n = 2$ and $n = 4$. Table 7 summarizes the results, including performance metrics and total training time (h) for each configuration across the Criteo and Avazu datasets.

Our observations indicate that increasing the number of components (n) results in a marginal improvement in performance metrics at the cost of total training time. For instance, with $n = 4$, the training time is approximately 1.29 times longer than with $n = 3$ and about 1.76 times longer than with $n = 2$. Based on these results, we choose $n = 3$ as the default configuration for OptFusion, as it offers a balanced trade-off between performance and efficiency.

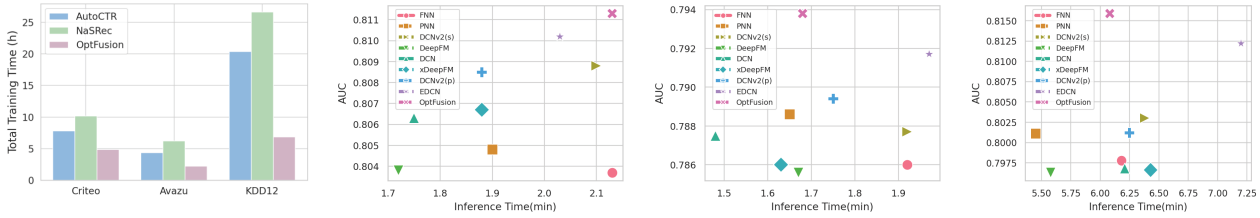
Table 7: Ablation study on the number of components.

Number	Criteo			Avazu		
	AUC	Logloss	Time (h)	AUC	Logloss	Time (h)
n=2	0.8112	0.4408	3.38h	0.7937	0.3716	1.47h
n=3	0.8113	0.4408	4.70h	0.7938	0.3715	1.98h
n=4	0.8115	0.4406	5.76h	0.7939	0.3717	2.67h

4.5 Case Study (RQ7)

This section uses a case study to investigate the selected connection and operation obtained from OptFusion on three datasets. The selected results, including both connection and operation, are shown in Figure 3. For better visualization, we only highlight the operation with the highest probability in OptFusion-soft, which is also the selected operation in OptFusion-hard.

Based on the result, we can make the following observation: First, the fusion searched on the Criteo dataset exhibits a preference for parallel structure, while on the Avazu and KDD12 datasets,



(a) Total Training Time on Three Datasets (b) Criteo Inference time (c) Avazu Inference time (d) KDD12 Inference time

Figure 2: Efficiency analysis across three datasets. The figure includes (a) total training time on three datasets and (b–d) inference efficiency analysis for the Criteo, Avazu, and KDD12 datasets.

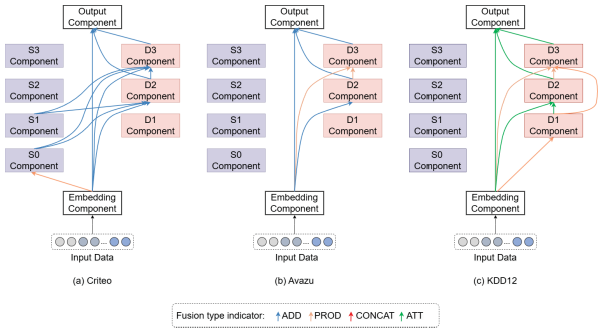


Figure 3: A case study of OptFusion on three datasets.

the learned architectures indicate a prevalent inclination towards stacked structure. Second, the output layer prefers to fuse connections from deep layers and the embedding layer. In addition, the embedding layer is connected to many other blocks. Third, the preference for fusion operation varies among datasets. ADD and PROD operations are the most common operations on the Criteo and Avazu datasets, while ATT and PROD are most frequently used on the KDD12 dataset. This case study provides valuable insights into the selected fusion by OptFusion, demonstrating the effectiveness and adaptability of OptFusion to different datasets. It may also provide design insights for future CTR models.

5 Related Work

5.1 Deep CTR Prediction Models

Most CTR models adopt two naive design fusion designs [35], parallel and stacked. Models with parallel fusion [5, 14, 32, 36, 37] leverage shallow and deep components that explicitly and implicitly model feature interactions, respectively. Fusion operations mainly are addition [5] or concatenation [36, 37]. Models with stacked fusion [6, 7, 15, 26, 37, 40] tend to stack the shallow components before the deep components with concatenation being the common fusion operation [26]. These models mainly advance CTR prediction by proposing various shallow components, such as inner product [26], factorization machine [5], outer product [6], convolutional operator [15], Hadamard product [40] and different customized layers [7, 14, 36, 37] or deep components, such as MLP [43] and Self-attention layer [32], to better model feature interactions.

Researchers also proposed methods with expert-designed fusion that are beyond parallel and stacked design. EDCN [3] performs a dense fusion strategy and captures the layer-wise interactive signals between the deep and shallow components. FinalMLP [24] proposes a Multi-Head Bilinear Fusion Ops as the fusion operation. EulerNet [34], on the other hand, explores feature interaction learning using Euler's formula, enabling adaptive and efficient fusion of feature interactions in CTR prediction models. However, these proposed solutions [3, 24] tend to consider the fusion learning problem under specific settings.

OptFusion differs itself by automatically learning connections and selecting operations. Many of the aforementioned methods can be considered as specific instances of the OptFusion framework.

5.2 Neural Architecture Search and its Applications in CTR Prediction

With the advancement of neural architecture search (NAS) [1, 8, 18, 19, 39], various methods have been proposed in CTR prediction [33], proving valuable for tasks such as determining appropriate embedding dimensions [10], conducting feature selection [17, 21], discovering beneficial feature interactions [16, 22], selecting integration function [11, 20], optimizing hyperparameters [13], designing comprehensive architectures for feature interaction modeling [25], or learning suitable embedding table [23]. Various techniques such as evolutionary approach [27], gradient approach [18], or reinforcement learning-based methods [44] are introduced to obtain suitable search results. Specifically, NAS techniques have also been adopted to search for suitable CTR model structures [31, 42]. Our work distinguishes itself from the existing research by addressing the challenge of fusion learning, a different problem in deep CTR models. The connection learning and operation selection among components are introduced as OptFusion's search space, enabling more efficient and effective model architectures for CTR prediction.

6 Conclusion

In this paper, we address the challenges of fusion learning in deep CTR prediction models and propose OptFusion, which automatically selects suitable fusion connections and fusion operations. OptFusion involves a one-shot learning algorithm designed to effectively conduct both tasks. The model is subsequently retrained with the learned architecture. Extensive experiments on three large-scale datasets demonstrate the superior performance of OptFusion in terms of efficiency and effectiveness. Several ablation studies investigate the configuration of OptFusion in improving prediction performance. Additionally, a case study on the connection and fusion operations further validates the efficacy of our approach in learning suitable architectures.

Ethical Consideration

In conducting our research and proposing the OptFusion approach, we have adhered to ethical considerations to ensure the integrity and social responsibility of our work. Our research primarily focuses on advancing the technical aspects of recommendation systems. Our experiment is based on public benchmarks and does not involve direct interactions with human subjects or the collection of personal data. As such, potential ethical concerns related to informed consent, privacy, and data handling are minimized.

Our work aims to contribute to the field of recommendation systems by addressing technical challenges associated with automatically learning suitable fusions in click-through rate prediction models. Throughout our research process, we have followed established research ethics guidelines and practices to ensure the accuracy, transparency, and rigor of our methods and results. We have also taken care to properly attribute prior works and provide appropriate citations to relevant sources to maintain academic integrity.

We acknowledge that while our research primarily concerns technical advancements, the deployment and application of recommendation systems in real-world scenarios may raise broader ethical considerations related to user privacy, fairness, and potential algorithmic biases. We recognize their significance and encourage researchers and practitioners to approach the deployment of recommendation systems with careful consideration of these ethical implications.

In summary, our research on OptFusion has been conducted with a commitment to upholding ethical standards within the scope of our technical contributions.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *CoRR* abs/1308.3432 (2013). arXiv:1308.3432 http://arxiv.org/abs/1308.3432

[2] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4 (dec 2015), 61. https://doi.org/10.1145/2532128

[3] Bo Chen, Yichao Wang, Zhirong Liu, Ruiming Tang, Wei Guo, Hongkun Zheng, Weiwei Yao, Muyu Zhang, and Xiuqiang He. 2021. Enhancing Explicit and Implicit Feature Interactions via Information Sharing for Parallel Deep CTR Models. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management*. ACM, Queensland, Australia, 3757–3766. https://doi.org/10.1145/3459637.3481915

[4] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016*. ACM, Boston, MA, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *26th International Joint Conference on Artificial Intelligence, IJCAI 2017*. ijcai.org, Melbourne, Australia, 1725–1731. https://doi.org/10.24963/IJCAI.2017/239

[6] Wei Guo, Ruiming Tang, Huifeng Guo, Jianhua Han, Wen Yang, and Yuzhou Zhang. 2019. Order-aware Embedding Neural Network for CTR Prediction. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019*. ACM, Paris, France, 1121–1124. https://doi.org/10.1145/3331184.3331332

[7] Tongwen Huang, Zhiqi Zhang, and Junlin Zhang. 2019. FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019*. ACM, Copenhagen, Denmark, 169–177. https://doi.org/10.1145/3298689.3347043

[8] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparameterization with Gumbel-Softmax. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France. https://openreview.net/forum?id=rkE3y85ee

[9] Olivier Chapelle Jean-Baptiste Tien, joycenv. 2014. Display Advertising Challenge. https://kaggle.com/competitions/criteo-display-ad-challenge

[10] Manas R. Joglekar, Cong Li, Mei Chen, Taibai Xu, Xiaoming Wang, Jay K. Adams, Pranav Khaitan, Jiahui Liu, and Quoc V. Le. 2020. Neural Input Search for Large Scale Recommendation Models. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, CA, USA, 2387–2397. https://doi.org/10.1145/3394486.3403288

[11] Farhan Khawar, Xu Hang, Ruiming Tang, Bin Liu, Zhenguo Li, and Xiuqiang He. 2020. AutoFeature: Searching for Feature Interactions and Their Architectures for Click-through Rate Prediction. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*. ACM, Ireland, 625–634. https://doi.org/10.1145/3340531.3411912

[12] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*. OpenReview.net, San Diego, CA, USA. http://arxiv.org/abs/1412.6980

[13] Yujun Li, Xing Tang, Bo Chen, Yimin Huang, Ruiming Tang, and Zhenguo Li. 2023. AutoOpt: Automatic Hyperparameter Scheduling and Optimization for Deep Click-through Rate Prediction. In *Proceedings of the 17th ACM Conference on Recommender Systems, RecSys 2023*. ACM, Singapore, Singapore, 183–194. https://doi.org/10.1145/3604915.3608800

[14] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*. ACM, London, UK, 1754–1763. https://doi.org/10.1145/3219819.3220023

[15] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, WWW 2019*. ACM, San Francisco, CA, USA, 1119–1129. https://doi.org/10.1145/3308558.3313497

[16] 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 *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, CA, USA, 2636–2645. https://doi.org/10.1145/3394486.3403314

[17] Dugang Liu, Chaohua Yang, Xing Tang, Yejing Wang, Fuyuan Lyu, Weihong Luo, Xiuqiang He, Zhong Ming, and Xiangyu Zhao. 2024. MultiFS: Automated Multi-Scenario Feature Selection in Deep Recommender Systems. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM 2024*. ACM, Merida, Mexico, 434–442. https://doi.org/10.1145/3616855.3635859

[18] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable Architecture Search. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, New Orleans, LA, USA. https://openreview.net/forum?id=S1eYHoC5FX

[19] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. 2018. Neural Architecture Optimization. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*. Curran Associates, Montreal, Canada, 7827–7838. https://proceedings.neurips.cc/paper/2018/hash/933670f1ac8ba969f32989c312faba75-Abstract.html

[20] Fuyuan Lyu, Xing Tang, Huifeng Guo, Ruiming Tang, Xiuqiang He, Rui Zhang, and Xue Liu. 2022. Memorize, Factorize, or be Naive: Learning Optimal Feature Interaction Methods for CTR Prediction. In *38th IEEE International Conference on Data Engineering, ICDE 2022*. IEEE, Kuala Lumpur, Malaysia, 1450–1462. https://doi.org/10.1109/ICDE53745.2022.00113

[21] Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing Feature Set for Click-Through Rate Prediction. In *Proceedings of the ACM Web Conference 2023, WWW 2023*. ACM, Austin, TX, USA, 3386–3395. https://doi.org/10.1145/3543507.3583545

[22] Fuyuan Lyu, Xing Tang, Dugang Liu, Chen Ma, Weihong Luo, Liang Chen, Xiuqiang He, and Xue (Steve) Liu. 2023. Towards Hybrid-grained Feature Interaction Selection for Deep Sparse Network. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*. Curran Associates, New Orleans, LA, USA. http://papers.nips.cc/paper_files/paper/2023/hash/9ab8da29b1eb3bec912a06e0879065cd-Abstract-Conference.html

[23] Fuyuan Lyu, Xing Tang, Hong Zhu, Huifeng Guo, Yingxue Zhang, Ruiming Tang, and Xue Liu. 2022. OptEmbed: Learning Optimal Embedding Table for Click-through Rate Prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. ACM, Atlanta, GA, USA, 1399–1409. https://doi.org/10.1145/3511808.3557411

[24] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, Washington, DC, USA, 4552–4560. https://doi.org/10.1609/AAAI.V37I4.25577

[25] Ze Meng, Jinnian Zhang, Yumeng Li, Jiancheng Li, Tanchao Zhu, and Lifeng Sun. 2021. A General Method For Automatic Discovery of Powerful Interactions In Click-Through Rate Prediction. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Canada, 1298–1307. https://doi.org/10.1145/3404835.3462842

[26] 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 *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Barcelona, Spain, 1149–1154. https://doi.org/10.1109/ICDM.2016.0151

[27] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka I. Leon-Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. 2017. Large-Scale Evolution of Image Classifiers. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017 (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, Sydney, NSW, Australia, 2902–2911. http://proceedings.mlr.press/v70/real17a.html

[28] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining*. IEEE Computer Society, Sydney, Australia, 995–1000. https://doi.org/10.1109/ICDM.2010.127

[29] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *16th International Conference on World Wide Web, WWW 2007*. ACM, Banff, Alberta, Canada, 521–530. https://doi.org/10.1145/1242572.1242643

[30] Michael S. Ryoo, A. J. Piergiovanni, Juhana Kangaspunta, and Anelia Angelova. 2020. AssembleNet++: Assembling Modality Representations via Attention Connections. In *Computer Vision - ECCV 2020 - 16th European Conference (Lecture Notes in Computer Science, Vol. 12365)*. Springer, Glasgow, UK, 654–671. https://doi.org/10.1007/978-3-030-58565-5_39

[31] Qingquan Song, Dehua Cheng, Hanning Zhou, Jiyan Yang, Yuandong Tian, and Xia Hu. 2020. Towards Automated Neural Interaction Discovery for Click-Through Rate Prediction. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. ACM, Virtual Event, CA, USA, 945–955. https://doi.org/10.1145/3394486.3403137

[32] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. AutoInt: Automatic Feature Interaction Learning via Self-Attentive Neural Networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019*. ACM, Beijing, China, 1161–1170. https://doi.org/10.1145/3357384.3357925

[33] Ruiming Tang, Bo Chen, Yejing Wang, Huifeng Guo, Yong Liu, Wenqi Fan, and Xiangyu Zhao. 2023. AutoML for Deep Recommender Systems: Fundamentals and Advances. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023*. ACM, Singapore, 1264–1267. https://doi.org/10.1145/3539597.3572729

[34] Zhen Tian, Ting Bai, Wayne Xin Zhao, Ji-Rong Wen, and Zhao Cao. 2023. EulerNet: Adaptive Feature Interaction Learning via Euler's Formula for CTR Prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Taipei, Taiwan, 1376–1385.

[35] Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Enhancing CTR Prediction with Context-Aware Feature Representation Learning. In *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, Madrid, Spain, 343–352. https://doi.org/10.1145/3477495.3531970

[36] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD'17 (ADKDD'17)*. Association for Computing Machinery, Halifax, NS, Canada, Article 12, 7 pages. https://doi.org/10.1145/3124749.3124754

[37] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *WWW '21: The Web Conference 2021*. ACM / IW3C2, Ljubljana, Slovenia, 1785–1797. https://doi.org/10.1145/3442381.3450078

[38] Zhiqiang Wang, Qingyun She, and Junlin Zhang. 2021. MaskNet: Introducing Feature-Wise Multiplication to CTR Ranking Models by Instance-Guided Mask. *CoRR* abs/2102.07619 (2021). arXiv:2102.07619 https://arxiv.org/abs/2102.07619

[39] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019*. Computer Vision Foundation / IEEE, Long Beach, CA, USA, 10734–10742. https://doi.org/10.1109/CVPR.2019.01099

[40] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017*. ijcai.org, Melbourne, Australia, 3119–3125. https://doi.org/10.24963/IJCAI.2017/435

[41] Kexin Zhang, Yichao Wang, Xiu Li, Ruiming Tang, and Rui Zhang. 2024. IncMSR: An Incremental Learning Approach for Multi-Scenario Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM 2024*. ACM, Merida, Mexico, 939–948. https://doi.org/10.1145/3616855.3635828

[42] Tunhou Zhang, Dehua Cheng, Yuchen He, Zhengxing Chen, Xiaoliang Dai, Liang Xiong, Feng Yan, Hai Li, Yiran Chen, and Wei Wen. 2023. NASRec: Weight Sharing Neural Architecture Search for Recommender Systems. In *Proceedings of the ACM Web Conference 2023, WWW 2023*. ACM, Austin, TX, USA, 1199–1207. https://doi.org/10.1145/3543507.3583446

[43] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data - - A Case Study on User Response Prediction. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016 (Lecture Notes in Computer Science, Vol. 9626)*. Springer, Padua, Italy, 45–57. https://doi.org/10.1007/978-3-319-30671-1_4

[44] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France. https://openreview.net/forum?id=r1Ue8Hcxg