CL4CTR: A Contrastive Learning Framework for CTR Prediction

Fangye Wang*
School of Computer Science
Fudan University, Shanghai, China
fywang18@fudan.edu.cn

Yingxu Wang*
School of Computer Science
Fudan University, Shanghai, China
yingxuwang20@fudan.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongsli@microsoft.com

Hansu Gu†
Seattle, United States
hansug@acm.org

Tun Lu*†
School of Computer Science
Fudan University, Shanghai, China
lutun@fudan.edu.cn

Peng Zhang*
School of Computer Science
Fudan University, Shanghai, China
zhangpeng_@fudan.edu.cn

Ning Gu*
School of Computer Science
Fudan University, Shanghai, China
ninggu@fudan.edu.cn

ABSTRACT

Many Click-Through Rate (CTR) prediction works focused on designing advanced architectures to model complex feature interactions but neglected the importance of feature representation learning, e.g., adopting a plain embedding layer for each feature, which results in sub-optimal feature representations and thus inferior CTR prediction performance. For instance, low frequency features, which account for the majority of features in many CTR tasks, are less considered in standard supervised learning settings, leading to sub-optimal feature representations. In this paper, we introduce self-supervised learning to produce high-quality feature representations directly and propose a model-agnostic Contrastive Learning for CTR (CL4CTR) framework consisting of three self-supervised learning signals to regularize the feature representation learning: contrastive loss, feature alignment, and field uniformity. The contrastive module first constructs positive feature pairs by data augmentation and then minimizes the distance between the representations of each positive feature pair by the contrastive loss. The feature alignment constraint forces the representations of features from the same field to be close, and the field uniformity constraint forces the representations of features from different fields to be distant. Extensive experiments verify that CL4CTR achieves the best performance on four datasets and has excellent effectiveness and compatibility with various representative baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

*Also Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China.
†Corresponding author.

WSDM '23, February 27–March 3, 2023, Singapore, Singapore.

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570372

KEYWORDS

Contrastive Learning, Representation Learning, CTR Prediction.

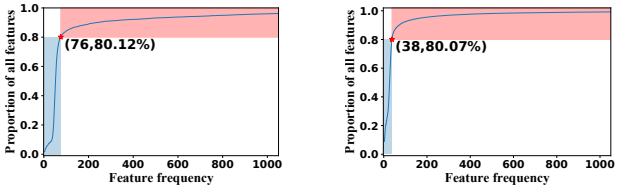
ACM Reference Format:
Fangye Wang, Yingxu Wang, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2023. CL4CTR: A Contrastive Learning Framework for CTR Prediction. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27–March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570372

1 INTRODUCTION

CTR prediction [7, 43], aiming to predict the probability of a given item being clicked, has been widely used in many applications, e.g., recommender systems [4] and computational advertising [18]. Recently, many methods [1, 33] achieved huge success by modeling complex feature interactions (FI). Following recent works [8, 13, 33], we categorize CTR prediction methods into two types: (1) traditional methods, such as logistic regression (LR) [27] and FM-based models [10, 26], can only model low-order feature interactions; and (2) deep-learning based methods, such as xDeepFM [17] and DCN-V2 [35], can further enhance the accuracy of CTR prediction by capturing high-order FI. In addition, many novel architectures (e.g., self-attention [16, 28], CIN [17], PIN [25]) have been proposed and widely deployed to capture sophisticated arbitrary-order FI.

Although successful in performance, many existing CTR prediction methods suffer from an inherent problem: *high frequency features have higher chances to be trained than low frequency features*, causing the representations of low frequency features to be sub-optimal. In Figure 1, we present the feature cumulative distributions of Frappe and ML-tag datasets. We can observe a clear "long tail" distribution of feature frequencies, e.g., bottom 80% of features appeared only 38 times or less in the ML-tag dataset. Since most CTR prediction models learn feature representations by the back-propagation [43], the low frequency features cannot be sufficiently trained due to less appearance, resulting in sub-optimal feature representations and thus sub-optimal CTR prediction performance.

Several prior works [19, 42] have also realized the importance of feature representation learning and proposed to deploy a weight



(a) Frappe (b) ML-tag

Figure 1: Cumulative distribution of feature frequencies. (38, 80.07%) indicates that features with feature frequencies less than or equal to 38 times account for 80.07% of all features.

learning module (i.e., FEN [42], Dual-FEN [19]) after the embedding layer which assigns weights for each feature to enhance their representations. However, the additional weighting modules will increase the model parameters and inference time. In addition, similar to FI-based methods, these methods only use the supervised learning signals to optimize feature representations from the plain embedding layer, which is not strong enough to produce accurate feature representations. Therefore, in this paper, we focus on directly learning accurate feature representations from the embedding layer without introducing additional weighting mechanisms, which is model-agnostic and has not been extensively studied.

In this paper, we seek to utilize self-supervised learning (SSL) to address the above issue, in which we design self-supervised learning signals as constraints to regularize the learned feature representations during the training process. As shown in Figure 2, we propose a novel framework called Contrastive Learning for Click Through Rate Prediction (**CL4CTR**), which consists of three key modules: CTR prediction model, contrastive module, and alignment&uniformity constraints. In detail, the CTR prediction model aims to predict the probability of items being clicked by a user, which can be replaced with most existing CTR models in the CL4CTR framework. In the contrastive module, we design three key components: (1) a data augmentation unit aiming to generate two different views for the output embedding as positive training pairs, which includes three different permutation approaches: random mask, feature mask, and dimension mask; (2) a feature interaction encoder aiming to learn compact FI representations based on the perturbed embeddings from the data augmentation unit; and (3) a task-oriented contrastive loss, which is designed to minimize the distance between the positive training pairs. In addition, we introduce two constraints: feature alignment and field uniformity, to facilitate contrastive learning. Feature alignment forces the representations of features from the same field to be as close as possible, and field uniformity forces the representations of features from different fields to be as distant as possible.

Our major contributions are summarized as follows:

- We propose a model-agnostic contrastive learning framework – CL4CTR, which can directly improve the quality of feature representations in an end-to-end manner.
- Considering the unique characteristics of CTR prediction tasks, we design three self-supervised learning signals: contrastive loss, feature alignment constraint and field uniformity constraint to improve contrastive learning performance.
- Extensive experiments on four datasets demonstrate that simply applying CL4CTR into FM [26] can outperform state-of-the-art methods. More importantly, CL4CTR shows high compatibility with existing methods, i.e., it can generally improve the performance of many representative baselines.

2 RELATED WORK

2.1 Deep CTR Prediction

According to the main focuses, recent CTR prediction works can be divided into two categories: FI-based methods [35, 43] and user interests modeling based methods [24]. Since our CL4CTR framework can be generally applied in FI-based models, we briefly summarize the FI-based works in this section. Most FI-based CTR prediction methods follow the common design paradigm: embedding layer, FI layer, and prediction layer. Some classical methods can only model fixed-order or low-order feature interactions. For instance, FM-based methods [14, 19, 26] model all pairwise interactions by using factorized parameters. Due to the importance of FI in the CTR prediction, many works focus on how to design novel structures for the FI layer to capture more informative and complicated feature interactions. Wide&Deep (WDL) [4] jointly trains the wide linear unit and Deep Neural Network (DNN) to combine memorization and generalization. DeepFM [7] comprises DNN and FM, and xDeepFM [17] additional proposes Compressed Interaction Network (CIN) based DeepFM to model high-order feature interaction explicitly. DCN [34] and DCN-V2 [35] explicitly and automatically use a cross-vector/cross-matrix network to improve the accuracy and efficiency of the DNN model. Furthermore, attention mechanism is one of the most effective structure to improve the performance and has been widely adopted for different purposes, e.g., AFM [39], Autoint [28], InterCTR [16], DCAP [3].

Notably, some works [13, 19, 42] attempt to improve the performance of CTR prediction by assigning different weights for features, in which they deploy a weight learning module to adjust the importance of feature representations after the embedding layer. However, these additional weighting modules may increase the model parameters and inference time. More importantly, these works only learn feature representations from a plain embedding layer, which is not strong enough to produce accurate feature representations as demonstrated in our experiments. The proposed CL4CTR can directly improve the quality of feature representations without requiring any additional modules after the embedding layer.

2.2 Self-supervised Learning

Recently, self-supervised learning has achieved remarkable success in learning powerful representations in many machine learning tasks [2, 6, 9, 15, 31, 40]. Contrastive Learning is one of the mainstream methods in SSL, which learns representations by attracting the positive sample pairs and repulsing the negative sample pairs [8]. Wang and Isola [36] identify two key properties related to the success of contrastive learning, i.e., alignment and uniformity. Alignment favors encoders that assign similar features to similar samples. Uniformity prefers a feature distribution that preserves maximal information.

In CTR prediction tasks, contrastive learning has not been extensively studied. Guo et al. [8] focus on sequential-based CTR tasks,

which apply interest-level contrastive losses to enhance feature embeddings. Pan et al. [23] propose an auxiliary AQCL loss that automatically leverages instance-instance similarity and instance-cluster similarity to regularize feature representations under the cold-start scenarios. Unlike them, our CL4CTR focuses on FI-based CTR prediction models, which can enhance the quality of feature representations by designing three SSL signals: contrastive loss, feature alignment constraint, and field uniformity constraint.

3 THE CL4CTR FRAMEWORK

3.1 CTR Prediction

CTR prediction is a binary classification task [23, 33]. Suppose a dataset for training CTR prediction model contains N instances $(\mathbf{x}, y)$, where $y \in \{0, 1\}$ (click or not) is the true label indicating user's click behaviors. Input instance $\mathbf{x}$ is usually multi-field tabular data record [8, 20], which contains F different fields and M features, as shown in Table 1. Recently, as shown in Figure 2(a), many CTR prediction models follow the common design paradigm [33, 37]: embedding layer, FI layer, and prediction layer.

Embedding layer. Generally, each input instance $\mathbf{x}_i$ is a sparse high-dimensional vector represented by a one-hot vector [18, 33]. And embedding layer transforms the sparse high-dimensional features $\mathbf{x}_i$ into a dense low-dimensional embedding matrix $\mathbf{E} = [\mathbf{e}^1; \mathbf{e}^2; ...; \mathbf{e}^F] \in \mathbb{R}^{F \times D}$, where D is the dimension size. Additionally, we use $\mathbb{E} = [\mathbf{E}_1, \mathbf{E}_2, ..., \mathbf{E}_F] \in \mathbb{R}^{M \times D}$ to represent all feature representations, where $\mathbf{E}_f$ is the subset representation of the f-th field $f \in \{1, 2, ..., F\}$. $|\mathbf{E}_f|$ is the number of features belonging to field f, and $M = \sum_{f=1}^{F} |\mathbf{E}_f|$.

Feature interaction layer. The FI layer usually contains various types of interaction operations to capture arbitrary-order feature interactions, such as MLP [4, 7], Cross Network [34], Cross Network2 [35] and transformer layer [16, 28], etc. We refer to these structures as feature interaction encoders, represented by $FI(\cdot)$. $FI(\cdot)$ can generate a compact feature interaction representation h_i based on embedding matrix $\mathbf{E}$.

Prediction layer. Finally, a prediction layer (usually a linear regression or MLP module) produces the final prediction probability $\sigma(\hat{y}_i) \in [0, 1]$ based on the compact representations h_i from the FI layer, where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function.

Finally, with the predicted label $\hat{y}_i$ and the true label y_i, the commonly adopted loss function of CTR models is as follows:

$$\mathcal{L}_{ctr} = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log \left(\sigma \left(\hat{y}_i \right) \right) + \left(1 - y_i \right) \log \left(1 - \sigma \left(\hat{y}_i \right) \right) \right). \quad (1)$$

Contrastive learning. As shown in Figure 2, in addition to the above components, we propose three contrastive learning signals: contrastive loss, feature alignment constraint and field uniformity constraint on top of the embedding layer to regularize the representation learning. Since these signals are not necessary during model inference, our method will not increase the inference time and the parameters of the underlying CTR prediction models.

3.2 Contrastive Module

Inspired by the success of SSL, we seek to deploy contrastive learning in the CTR prediction tasks to generate high-quality feature representations. As illustrated in Figure 2(b), the contrastive module consists of three major components: a data augmentation unit, a FI encoder, and a contrastive loss function. In the data augmentation unit, we propose three different task-oriented posterior embedding augmentation techniques to generate positive training pairs, i.e., two different views of each feature embedding. Then we feed the two perturbed embeddings to the same FI encoder to generate two compressed feature representations. Finally, the contrastive loss is applied to minimize the distance between the two compressed feature representations.

Table 1: An example of multi-field tabular data for CTR prediction. Each row represents an input instance and each column indicates a field. Moreover, each field contains multiple features, but each feature only belongs to one field.

user_id	item_id	gender	city	daytime	...	click
25c83c98	c5c50484	female	5	1	...	0
7e0ccccf	0b153874	male	10	5	...	1
de7995b8	e51ddf94	male	32	6	...	1
1f89b562	f0cf0024	female	4	2	...	1
1f89b562	a3397841	female	4	8	...	0

3.2.1 Data Augmentation via Output Perturbation. Data augmentation has shown great potential in improving the performance of feature representations in SSL [41]. Different and well-designed augmentation approaches have been proposed and used to construct different views of the same input instance. For example, in the scenarios of sequential recommendation, three widely used augmentation methods are item masking, reordering, and cropping [41]. However, these methods are designed to augment behavior sequences and are not appropriately deployed in FI-based CTR prediction models. Hence, we firstly propose three task-oriented augmentation approaches, which aim to perturb feature embeddings for FI-based models. As shown in Figure 2(d), we use the function $\hat{\mathbf{E}} = \mathrm{g}(\mathbf{E})$ to represent data augmentation process.

Random Mask. Firstly, we introduce the random mask method, which is analogous to Dropout [11]. This method randomly masks some elements in initial embedding $\mathbf{E}$ with a certain probability p. The random mask is generated as follows:

$$\hat{\mathbf{E}} = \mathrm{g}_r(\mathbf{E}) = \mathbf{E} \cdot \mathbf{I}, \mathbf{I} \sim \text{Bernoulli}(p) \in \mathbb{E}^{F \times D}. \quad (2)$$

Bernoulli$(\cdot)$ is the Bernoulli distribution, and $\mathbf{I}$ is a matrix of Bernoulli random variables each of which has probability p of being 1.

Feature Mask. Motivated by prior works [18, 28], we propose to mask the feature information in the initial embedding, where the feature mask can be generated as follows:

$$\hat{\mathbf{E}} = \mathrm{g}_f(\mathbf{E}) = [\hat{\mathbf{e}}^1; \hat{\mathbf{e}}^2; ...; \hat{\mathbf{e}}^F], \hat{\mathbf{e}}^f = \begin{cases} \mathbf{e}^f, & t \notin \mathcal{T} \\ [\text{mask}], & t \in \mathcal{T} \end{cases}, \quad (3)$$

where we set a proportion p of features $\mathcal{T} = (t_1, t_2, ..., t_{L_F})$ with the length $L_F = \lfloor p * F \rfloor$. t_f is the index of feature in $\mathbf{E}$. If one feature is masked, then the representation of this feature will be replaced with [mask], which is a zero vector.

Dimension Mask. The dimensions of feature representations affect the effectiveness of deep learning models. Inspired by FED [44], which attempts to improve prediction performance by capturing dimension relations, we propose to perturb the initial embedding

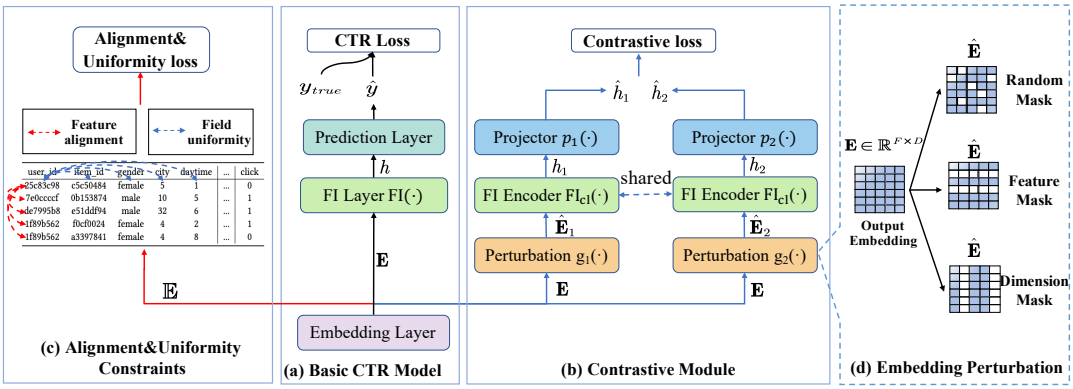


Figure 2: Architecture of the CL4CTR framework. CL4CTR including three components: (a) a basic CTR model; (b) a contrastive module; (c) alignment & uniformity constraints. In contrastive module, we design (d) three embedding perturbation methods.

by replacing specific proportions of dimensional information of feature representations, which can be described as follows:

$$\hat{\mathbf{E}} = g_d(\mathbf{E}) = [d\mathbf{e}^1; d\mathbf{e}^2; ...; d\mathbf{e}^F], d \sim \text{Bernoulli}(p) \in \mathbb{R}^D, \quad (4)$$

where d is a vector of Bernoulli random variables, each of which has a probability p of being 1.

During the training process, we select one of the above mask methods to generate two perturbed embedding $\hat{\mathbf{E}}_1$ and $\hat{\mathbf{E}}_2$, where $\hat{\mathbf{E}}_1 = g(\mathbf{E})$ and $\hat{\mathbf{E}}_2 = g(\mathbf{E})$ in Figure 2(b). More analyses about the effectiveness of different mask methods are showed in Section 4.3.

3.2.2 Feature Interaction Encoder. We utilize a shared FI encoder to extract feature interaction information from the two perturbed embeddings $\hat{\mathbf{E}}_1$ and $\hat{\mathbf{E}}_2$ as follows:

$$h_1 = FI_{cl}(\hat{\mathbf{E}}_1), h_2 = FI_{cl}(\hat{\mathbf{E}}_2). \quad (5)$$

$FI_{cl}(\cdot)$ represents FI encoder function, and h_1, h_2 are two compressed representations generated from two perturbed embeddings.

Notably, any FI encoder can be deployed in our CL4CTR, such as cross-network [35], self-attention [28], and bi-interaction [10], as described in Section 3.1. Specifically, we select the Transformer layer [30] as our primary FI encoder, which is widely used to extract vector-level relationships between features [3, 28, 40].

Additionally, we find that the dimensions of compressed representations (h_1, h_2) generated by some FI encoders (e.g., cross network [35], PIN [25]) could be huge, e.g., over thousands when field F is large, which produce adversely impacts on the training stability. Hence, we utilize a projection function to reduce the dimensions of representations from FI encoder to D as follows:

$$\hat{h}_1 = p_1(h_1), \hat{h}_2 = p_2(h_2). \quad (6)$$

The projection function $p(\cdot)$ is a single layer MLP.

3.2.3 Contrastive Loss Function. Finally, a contrastive loss function is applied to minimize the expected distance between the above two perturbed representations as follows:

$$\mathcal{L}_{cl} = \frac{1}{B} \sum_{i=1}^{B} \left\| \hat{h}_{i,1} - \hat{h}_{i,2} \right\|_2^2. \quad (7)$$

B is the batch size and $||\cdot||_2^2$ denotes the ℓ_2 distance.

3.3 Feature Alignment and Field Uniformity

To ensure low frequency features and high-frequency features be trained equally, a naive way is to increase the frequency of low frequency features or reduce the frequency of high-frequency features during training. Inspired by previous works [32, 36] in other areas (CV, NLP), which can achieve similar goal by introducing two critical properties, named the alignment and uniformity constraints, but they need to construct positive and negative sample pairs to optimize the two constraints. In CTR prediction tasks, we find that features in the same field are analogous to positive sample pairs, and features of different fields are analogous to negative sample pairs. Thus, we propose two new properties for contrastive learning in CTR prediction, named feature alignment and field uniformity, which can regularize feature representations during training process. Specifically, feature alignment pulls the representations of features from the same field to be as close as possible. In contrast, field uniformity pushes representations of features from different fields to be as distant as possible.

3.3.1 Feature Alignment. Firstly, we introduce the feature alignment constraint, which aims to minimize the distance between features from the same field. Intuitively, by adding a feature alignment constraint, the representations of features in the same field should be more closely distributed in the low-dimensional space. Formally, the loss function of feature alignment is as follows:

$$\mathcal{L}_a = \sum_{f=1}^{F} \sum_{\mathbf{e_i}, \mathbf{e_j} \in \mathbf{E}_f} \|\mathbf{e_i} - \mathbf{e_j}\|_2^2, \quad (8)$$

where $\mathbf{e_i}$ and $\mathbf{e_j}$ are two features from the same field, and $\mathbf{E}_f$ is the subset features of field f.

3.3.2 Field Uniformity. The relationships among different fields have not been extensively studied in existing CTR prediction methods. For instance, FFM [14] learns field-aware representation for each feature, and NON [20] extracts intra-field information, but their techniques cannot be directly applied in contrastive learning. Differently, we introduce field uniformity to optimize feature representation directly, which minimizes the similarity between features belonging to different fields. The loss function of field uniformity

Table 2: Dataset statistics.

Datasets	Positive	#Training	#Validation	#Test	#Features	#Fields
Frappe	33%	202K	58K	29K	5K	10
ML-tag	33%	1,404K	401K	201K	90K	3
ML-1M	57.5%	800K	100K	100K	10K	5
SafeDriver	3.64%	476K	59K	59K	600	57

is formally defined as follows:

$$\mathcal{L}_u = \sum_{\substack{\mathbf{e_i} \in \mathbf{E}_f \\ 1<=f<=F}} \sum_{\mathbf{e_j} \in (\mathbb{E}-\mathbf{E}_f)} sim(\mathbf{e_i}, \mathbf{e_j}). \tag{9}$$

Similar to the other approaches [41, 46], we use cosine similarity to regularize negative sample pairs, i.e., $sim(\mathbf{e_i}, \mathbf{e_j}) = {\mathbf{e_i}}^T \mathbf{e_j} / \|\mathbf{e_i}\| \|\mathbf{e_j}\|$. Other similarity functions can also be used here. $\mathbb{E} - \mathbf{E}_f$ contains all features except those from field f.

In both feature alignment and field uniformity constraints, we find that low frequency features and high frequency features have equal chances to be considered. Therefore, the suboptimal representation issue for low frequency features can be largely alleviated when the two constraints are introduced during training.

3.4 Multi-task Training

To integrate the CL4CTR framework into the scenarios of CTR prediction, we adopt a multi-task training strategy to jointly optimize these three auxiliary SSL losses and the original CTR prediction loss in an end-to-end manner. Thus the final objective function can be formulated as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{ctr} + \alpha \cdot \mathcal{L}_{cl} + \beta \cdot (\mathcal{L}_a + \mathcal{L}_u), \tag{10}$$

where α and β are the hyper-parameters to control the strengths of contrastive loss and feature alignment and field uniformity loss.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Datasets. We evaluate CL4CTR on four popular datasets: **Frappe**[1] [10], **ML-tag**[2] [10], **SafeDriver**[3] [12] and **ML-1M**[4] [3]. The statistics of the four datasets are presented in Table 2. NFM [10] and AFM [39] have strictly split Frappe and ML-tag to training, validation, and testing by 7:2:1, and we directly follow their settings. For SafeDriver and ML-1M, following [12] and [3], we randomly split the instances by 8:1:1. Detailed descriptions of those datasets can be found in the links or references.

4.1.2 Compared Methods. To evaluate the proposed CL4CTR framework, we compare its performance with four classes of representative CTR methods [21]. 1) First-order method that is a weighted sum of raw features, including **LR**; 2) FM-based methods that consider second-order FI, including **FM** [26], **FwFM** [22], **IFM** [42], and **FmFM** [29]; 3) Approaches that model higher-order FI, including **CrossNet** [34], **IPNN** [25], **OPNN** [25], **FINT** [45], and **DCAP** [3]; 4) Ensemble methods or multi-tower structures, including **WDL** [4], **DCN** [34], **DeepFM** [7], **xDeepFM** [17], **FiBiNET** [13], **AutoInt+** [28], **AFN+** [5], **TFNET** [38], **FED** [44], and **DCN-V2** [35]. The proposed CL4CTR framework is model-agnostic. For simplicity, a base model $\boldsymbol{M}$ equipped with CL4CTR is represented as $CL4CTR_{\boldsymbol{M}}$. We choose FM [26] as the basic model to verify the effectiveness of CL4CTR, which only models second-order FI and has no additional parameters except for feature representations. **Therefore, the performance boost of $CL4CTR_{FM}$ directly reflects the quality of the feature representations.** CL4CTR only helps the base CTR models training and does not add any operation or parameter to the inference process.

4.1.3 Evaluation Metrics. To evaluate the performance of all methods, we adopt the commonly-used **AUC** (Area Under ROC) and **Logloss** (cross entropy) as the metrics. Notably, a slightly higher AUC or a lower Logloss at **0.001**-level can be considered significant for CTR prediction [1, 13, 17, 20, 35].

4.1.4 Implementation Details. For fair comparisons, we implement all the models with Pytorch[5] and optimize all models with Adam. The embedding size is set to 64 for Frappe and ML-tag and 20 for ML-1M and SafeDriver, respectively. Meanwhile, the batch size is fixed to 1024. the learning rate is 0.01 for SafeDriver and 0.001 for other datasets. As for the models including DNN in the prediction layer, we adopt the same structure {400,400,400,1}. All the activation functions are ReLU, and the dropout rate is 0.5. We perform early stopping according to AUC on the validation set to avoid overfitting. We also implement the Reduce-LR-On-Plateau scheduler to reduce the learning rate by a factor of 10 when the given metric stops improving within four continuous epochs. Each experiment is repeated 5 times, and the average results are reported. In CL4CTR, $FI_{cl}(\cdot)$ adopts three transformer layers. And we use hyper-parameters: α=1, β=0.01 in the final loss function.

4.2 Overall Comparison

In this section, we compare the performances of $CL4CTR_{FM}$ with the state-of-the-art (SOTA) CTR prediction models. Table 3 shows the experimental results of all compared models over four datasets.

It can be observed that LR and FM achieve the worst performance compared with other baselines, which indicates that shallow models are insufficient for CTR prediction. Other FM-based models improve FM by introducing field importance mechanism (e.g., FwFM [22] and IFM [42]) or deploying a novel field-pair matrix approach (e.g., FmFM [29]). Generally, deep-learning based models (e.g., DeepFM [7], DCN [34], DCN-V2 [35]), which combine high-order feature interactions with well-designed feature interaction structures, achieve better performance than FM.

$CL4CTR_{FM}$ consistently performs better than all baselines on all datasets. Furthermore, $CL4CTR_{FM}$ significantly outperforms the strongest baseline DCN-V2 [35] by 0.13%, 0.11%, 0.39% and 0.67% in terms of AUC (16.10%, 8.41%, 0.64% and 1.46% in terms of Logloss) on Frappe, ML-tag, ML-1M, and SafeDriver respectively. Additionally, we find that the improvements on Logloss are more remarkable than those on AUC, indicating that CL4CTR enables us to predict the true click probability effectively. Meanwhile, $CL4CTR_{FM}$ shows

[1] https://www.baltrunas.info/context-aware/frappe
[2] https://grouplens.org/datasets/movielens/
[3] https://www.kaggle.com/c/porto-seguro-safe-driver-prediction
[4] https://grouplens.org/datasets/movielens/1m/

[5] The code of CL4CTR is available here: https://github.com/cl4ctr/cl4ctr

Table 3: Overall accuracy comparison in the four datasets. ΔAUC and $\Delta Logloss$ indicate averaged performance boost compared with DCN-V2. *RelaImp* denotes the relative improvements compared with the strongest baseline. Bold scores are the best performance, while underlined scores are the second best. Improvements over baselines are statistically significant with p<0.01.

Model Class	Datasets Model	Frappe AUC	Frappe Logloss	ML-tag AUC	ML-tag Logloss	ML-1M AUC	ML-1M Logloss	SafeDriver AUC	SafeDriver Logloss	ΔAUC ↑	$\Delta Logloss$ ↓
First-order	LR	0.9331	0.2894	0.9348	0.2960	0.7899	0.5417	0.6244	0.1622	-3.35%	0.0572
Second-Order	FM	0.9746	0.1856	0.9488	0.2595	0.8023	0.5332	0.6301	0.1538	-1.22%	0.0179
	FwFM	0.9756	0.1784	0.9582	0.2531	0.8046	0.5281	0.6335	0.1532	-0.74%	0.0131
	IFM	0.9771	0.1581	0.9515	0.2497	0.8080	0.5286	0.6353	0.1526	-0.70%	0.0071
	FmFM	0.9801	0.1682	0.9552	0.2493	0.8093	0.5264	0.6378	0.1518	-0.39%	0.0088
High-Order	CrossNet	0.9800	0.1658	0.9549	0.2480	0.8114	0.5218	0.6336	0.1517	-0.50%	0.0067
	IPNN	0.9809	0.1604	0.9607	0.2295	0.8110	0.5190	0.6373	0.1521	-0.19%	0.0001
	OPNN	0.9799	0.1683	0.9599	0.2421	0.8112	0.5185	0.6375	0.1519	-0.22%	0.0051
	FINT	0.9807	0.1578	0.9557	0.2430	0.8123	0.5192	0.6349	0.1522	-0.38%	0.0029
	DCAP	0.9801	0.1612	0.9560	0.2428	0.8130	0.5171	0.6390	0.1512	-0.20%	0.0030
Ensemble	WDL	0.9770	0.1783	0.9599	0.2660	0.8093	0.5226	0.6353	0.1525	-0.44%	0.0110
	DCN	0.9788	0.1621	0.9550	0.2472	0.8125	0.5175	0.6379	0.1514	-0.32%	0.0044
	DeepFM	0.9780	0.1732	0.9586	0.2551	0.8061	0.5259	0.6318	0.1529	-0.69%	0.0117
	xDeepFM	0.9799	0.1750	0.9604	0.2472	0.8082	0.5244	0.6403	0.1515	-0.19%	0.0094
	FiBiNET	0.9793	0.1707	0.9548	0.2532	0.8032	0.5313	0.6391	0.1505	-0.56%	0.0113
	AutoInt+	0.9783	0.1762	0.9535	0.2562	0.8099	0.5219	0.6310	0.1516	-0.73%	0.0114
	AFN+	0.9786	0.1637	0.9561	0.2468	0.8041	0.5304	0.6374	0.1517	-0.58%	0.0080
	TFNet	0.9798	0.1708	0.9527	0.2551	0.8099	0.5212	0.6387	0.1533	-0.41%	0.0100
	FED	0.9791	0.1606	0.9557	0.2465	0.8128	0.5184	0.6369	0.1534	-0.33%	0.0046
	DCN-V2	0.9803	0.1595	0.9610	0.2330	0.8132	0.5169	0.6406	0.1510	-	-
Ours	$CL4CTR_{FM}$	**0.9822**	**0.1324**	**0.9621**	**0.2102**	**0.8164**	**0.5136**	**0.6449**	**0.1483**	0.34%	-0.0140
	RelaImp	0.13%	16.10%	0.11%	8.41%	0.39%	0.64%	0.67%	1.46%	-	-

strong generalization ability on all datasets, where Table 3 shows the averaged performance boost (ΔAUC and ΔLogloss). Notably, most SOTA CTR prediction models design complex networks to produce advanced feature representations and useful feature interactions to improve the performance. However, our CL4CTR only helps FM to learn accurate feature representations from the embedding layer with contrastive learning instead of introducing extra modules. The improvement of our CL4CTR verifies the necessity of learning accurate feature representations in CTR prediction tasks.

4.3 Ablation Study

4.3.1 Compatibility Analysis. To verify the compatibility of CL4CTR, we deploy it into other SOTA models, such as DeepFM [7], Autoint+ [28], and DCN-V2 [35]. The results are shown in Table 4.

Firstly, learning feature representation with CL4CTR can significantly improve the performance of CTR prediction. Applied with CL4CTR, the performance of base models is remarkably boosted, which confirms our hypothesis of improving the performance of CTR prediction models by improving the quality of the feature representations and demonstrates the effectiveness of CL4CTR. In addition, the experimental results show that learning high-quality feature representations is at least as important as designing advanced FI techniques. Modeling complex feature interactions can improve the performance of CTR models when these models leverage supervised signal for training, which explains why FI-based models outperform FM. However, after introducing self-supervised signals into CTR models for learning high-quality feature representations, FM can achieve the best performance compared with other models deployed with CL4CTR. The possible reason is that both supervised and self-supervised learning are directly and only optimizing the parameters in feature representations in FM without disturbing by other parameters.

Table 4: Compatibility study of CL4CTR.

Model	Frappe AUC	Frappe Logloss	ML-1M AUC	ML-1M Logloss	SafeDriver AUC	SafeDriver Logloss
FM	0.9746	0.1856	0.8023	0.5332	0.6244	0.1622
$CL4CTR_{FM}$	**0.9822**	**0.1324**	**0.8164**	**0.5136**	**0.6449**	**0.1483**
FwFM	0.9756	0.1784	0.8046	0.5281	0.6335	0.1532
$CL4CTR_{FwFM}$	**0.9815**	**0.1532**	**0.8118**	**0.5192**	**0.6421**	**0.1487**
DeepFM	0.9780	0.1732	0.8061	0.5259	0.6318	0.1529
$CL4CTR_{DeepFM}$	**0.9813**	**0.1677**	**0.8113**	**0.5194**	**0.6381**	**0.1504**
Autoint+	0.9783	0.1762	0.8099	0.5219	0.6310	0.1516
$CL4CTR_{Autoint+}$	**0.9802**	**0.1684**	**0.8122**	**0.5174**	**0.6402**	**0.1506**
DCN	0.9788	0.1621	0.8125	0.5170	0.6379	0.1514
$CL4CTR_{DCN}$	**0.9808**	**0.1566**	**0.8164**	**0.5125**	**0.6415**	**0.1494**
DCN-V2	0.9803	0.1595	0.8132	0.5169	0.6406	0.1510
$CL4CTR_{DCN-V2}$	**0.9812**	**0.1549**	**0.8144**	**0.5153**	**0.6411**	**0.1497**

4.3.2 Data Augmentation Approaches. To verify the effectiveness of our proposed data augmentation methods, we change the augmentation methods in the contrastive module and fix other settings for a fair comparison. Furthermore, we select different baseline models and deploy CL4CTR into them to compare their performance under this setting. Table 5 shows the experimental results.

The random mask method achieves the best performance in most cases. We think the random mask is more moderated than feature mask and dimension mask because it omits element information. Additionally, the FwFM model achieves the best performance with

Table 5: Impact of data augmentation methods.

Base model	Variants	Frappe		SafeDriver	
		AUC	Logloss	AUC	Logloss
FM	Base	0.9746	0.1856	0.6244	0.1622
	Random	**0.9822**	**0.1324**	**0.6449**	**0.1483**
	Feature	0.9814	0.1328	0.6303	0.1539
	Dimension	0.9816	0.1334	0.6404	0.1505
FwFM	Base	0.9756	0.1784	0.6335	0.1532
	Random	0.9815	0.1532	0.6421	0.1487
	Feature	**0.9822**	**0.1513**	0.6384	0.1483
	Dimension	0.9811	0.1465	**0.6455**	**0.1508**
DeepFM	Base	0.9780	0.1817	0.6318	0.1529
	Random	**0.9813**	**0.1677**	**0.6381**	**0.1504**
	Feature	0.9798	0.1750	0.6341	0.1522
	Dimension	0.9804	0.1697	0.6353	0.1514
DCN	Base	0.9788	0.1611	0.6379	0.1514
	Random	**0.9808**	**0.1566**	**0.6415**	**0.1494**
	Feature	0.9804	0.1601	0.6409	0.1508
	Dimension	0.9803	0.1573	0.6411	0.1504

Table 6: Impact of different FI encoder $FI_{cl}(\cdot)$.

Base model	FI Encoder	Frappe		ML-1M	
		AUC	Logloss	AUC	Logloss
FM	Base	0.9746	0.1856	0.8023	0.5332
	DNN	0.9804	0.1404	**0.8177**	**0.5123**
	Transformer	**0.9822**	**0.1324**	0.8164	0.5136
	CrossNet2	0.9801	0.1438	0.8170	0.5143
FwFM	Base	0.9756	0.1784	0.8046	0.5281
	DNN	0.9809	**0.1504**	0.8064	0.5264
	Transformer	0.9815	0.1532	**0.8118**	**0.5192**
	CrossNet2	**0.9822**	0.1675	0.8102	0.5231
DeepFM	Base	0.9780	0.1732	0.8061	0.5259
	DNN	0.9804	0.1710	0.8101	0.5206
	Transformer	**0.9813**	**0.1704**	**0.8113**	**0.5194**
	CrossNet2	0.9791	0.1719	0.8109	0.5202
DCN-V2	Base	0.9803	0.1595	0.8132	0.5169
	DNN	0.9807	0.1573	**0.8151**	**0.5144**
	Transformer	**0.9812**	**0.1549**	0.8144	0.5153
	CrossNet2	0.9804	0.1588	0.8141	0.5155

the feature mask method on Frappe; in contrast, it achieves the best performance with the dimension mask method on SafeDriver, demonstrating that our proposed augmentation methods are effective and can be used in different baseline models and datasets.

4.3.3 FI Encoder. In the contrastive module, the FI encoder also affects the performance of CL4CTR, as different structures of the FI encoder extract different information. For instance, transformer layer can model high-order feature interactions in feature-level [3, 28] explicitly, however, CrossNet2 [35] focuses on modelling bounded-degree feature interactions in element-level explicitly. DNN is a common and widely used structure in most CTR models for modeling bit-level feature relationships implicitly. We select the above three representative structures as FI encoders and verify their performance. Notably, we adopt three layers structure as reported in their paper. Table 6 shows the experimental results.

It can be observed that CL4CTR can consistently improve the performance of these baseline models with different FI encoders. In addition, since different FI encoders extract different information based on specific base models and datasets, the performance of these models is different. However, CL4CTR with transformer layers achieves the best performance in most cases since the transformer layer is a more effective solution than others.

Table 7: Impact of SSL signals in the loss function.

Model	Loss Function	Frappe		ML-1M	
		AUC	Logloss	AUC	Logloss
FM	$\mathcal{L}_{ctr}$	0.9746	0.1856	0.8023	0.5332
	+ $\mathcal{L}_{cl}$	0.9794	0.1485	0.8102	0.5230
	+ $(\mathcal{L}_a + \mathcal{L}_u)$	0.9812	0.1455	0.8139	0.5175
	+ $\mathcal{L}_{cl}$ + $(\mathcal{L}_a + \mathcal{L}_u)$	**0.9822**	**0.1324**	**0.8164**	**0.5136**
FwFM	$\mathcal{L}_{ctr}$	0.9756	0.1784	0.8046	0.5281
	+ $\mathcal{L}_{cl}$	0.9785	0.1553	0.8109	0.5229
	+ $(\mathcal{L}_a + \mathcal{L}_u)$	0.9812	0.1536	0.8098	0.5252
	+ $\mathcal{L}_{cl}$ + $(\mathcal{L}_a + \mathcal{L}_u)$	**0.9815**	**0.1532**	**0.8118**	**0.5192**
DeepFM	$\mathcal{L}_{ctr}$	0.9780	0.1817	0.8061	0.5259
	+ $\mathcal{L}_{cl}$	0.9794	0.1701	0.8094	0.5235
	+ $(\mathcal{L}_a + \mathcal{L}_u)$	0.9784	0.1791	0.8103	0.5214
	+ $\mathcal{L}_{cl}$ + $(\mathcal{L}_a + \mathcal{L}_u)$	**0.9813**	**0.1677**	**0.8113**	**0.5194**
DCN	$\mathcal{L}_{ctr}$	0.9788	0.1611	0.8125	0.5170
	+ $\mathcal{L}_{cl}$	0.9802	0.1585	0.8138	0.5150
	+ $(\mathcal{L}_a + \mathcal{L}_u)$	0.9792	0.1600	0.8129	0.5188
	+ $\mathcal{L}_{cl}$ + $(\mathcal{L}_a + \mathcal{L}_u)$	**0.9808**	**0.1566**	**0.8164**	**0.5125**

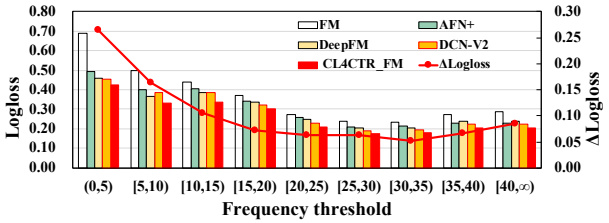


Figure 3: Improvement vs. feature frequency.

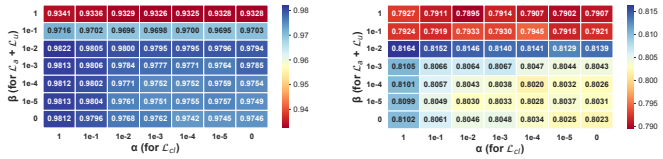
4.3.4 Loss Function. In this section, we evaluate the effectiveness of self-supervised learning signals (i.e., $\mathcal{L}_{cl}$, $\mathcal{L}_a$, $\mathcal{L}_u$) by eliminating them from CL4CTR respectively. We still regard $\mathcal{L}_a$ and $\mathcal{L}_u$ as a whole part. The experimental results are shown in Table 7.

Firstly, we can find that each self-supervised learning signal deployed in baseline models can improve their performance. In addition, by comparing the contrastive loss and alignment&uniformity constraints individually, we conclude that they play different roles in different datasets and baseline models. Specifically, FM with $\mathcal{L}_a$ and $\mathcal{L}_u$ perform better than FM with $\mathcal{L}_{cl}$; in contrast, DCN with $\mathcal{L}_a$ and $\mathcal{L}_u$ perform worse than DCN with $\mathcal{L}_{cl}$, which verifies our hypothesis. Furthermore, all experiments achieve the best performance when $\mathcal{L}_{cl}$, $\mathcal{L}_a$, and $\mathcal{L}_u$ are deployed simultaneously.

Compared with individual SSL signals, we find that training with all of them can always achieve the best performance. Furthermore, adopting $\mathcal{L}_a$ and $\mathcal{L}_u$ in FM consistently outperforms adopting $\mathcal{L}_{cl}$ in FM on two datasets evaluated by Logloss, which indicates that inducing feature alignment and field uniformity into CTR prediction models enables us to predict probabilities closer to the true label.

4.4 Feature Frequency Analysis

To verify the effects of feature frequency on different models, we divide the test set of ML-tag according to feature frequency and calculate the corresponding Logloss, where $\Delta Logloss$ represents the improvement over the base FM model after applying CL4CTR. Figure 3 shows the experimental results.



(a) The AUC of Frappe (b) The AUC of ML-1M

Figure 4: Performance of $CL4CTR_{FM}$ w.r.t. different weights assigned to three SSL signals: α for $\mathcal{L}_c$, β for $\mathcal{L}_a$ and $\mathcal{L}_u$.

Firstly, the low frequency features adversely affect the accuracy of the base model. Specifically, we show the performance of FM, three SOTA models (AFN+, DeepFM, DCN-V2), and $CL4CTR_{FM}$ in different frequency ranges. It can be observed that all models perform the worst when the input subset contains low frequency features. With the increasing of feature frequency, the performance of all models improves consistently. When the feature frequency is over 20, the performance of all models becomes stable. Figure 3 confirms our hypothesis that only using the back-propagation to learn the representations of low frequency features with a single supervised signal cannot achieve optimal performance.

Secondly, CL4CTR can effectively alleviate the negative effects caused by low frequency features and keep achieving the best performance among different feature frequency ranges. By applying the alignment&uniformity constraints, we ensure the low frequency features can be optimized in each back-propagation process with equal chances to high frequency features. Additionally, the contrastive module can also improve the quality of the representations of all features, including both low and high frequency features.

4.5 Hyper-parameter Analysis

4.5.1 Impact of the Weights in the Loss Function. We further investigate the impact of different weights (α and β in Equation 10). We tune both α and β from {1, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 0}. We keep other settings fixed for fair comparison. Figure 4 shows the experimental results. Additionally, the trend of Logloss is consistent with AUC on those two datasets.

Overall, CL4CTR achieves the best performance when α is 1, and β is 1e-2 for Frappe and ML-1M datasets. Specifically, the performance of CL4CTR deteriorates when α is less than 1. In addition, when β is over 1e-2 (i.e., 1 or 1e-1), the performance of CL4CTR is significantly reduced. Meanwhile, CL4CTR performs worse with lower α and β (lower right corners).

4.5.2 Mask Proportion. We change the mask proportion p within the range (0, 1) with a step size of 0.1. Note that the mask proportion is only applied in $\mathcal{L}_{cl}$. Figure 5 shows the results.

For models with $\mathcal{L}_{cl}$, their performance shows similar trends on Frappe and ML-1M. When the mask proportion is around 0.4 or 0.5, $CL4CTR_{FM}$ can achieve the best performance. Specifically, the model performance decreases slightly when smaller mask proportions (i.e., 0.1 to 0.3) are chosen. When mask proportion is over 0.5, the model performance decreases consistently, which is because the FI encoders only use a small percentage of information to produce valid interaction representations for calculating contrastive loss with higher mask proportions.

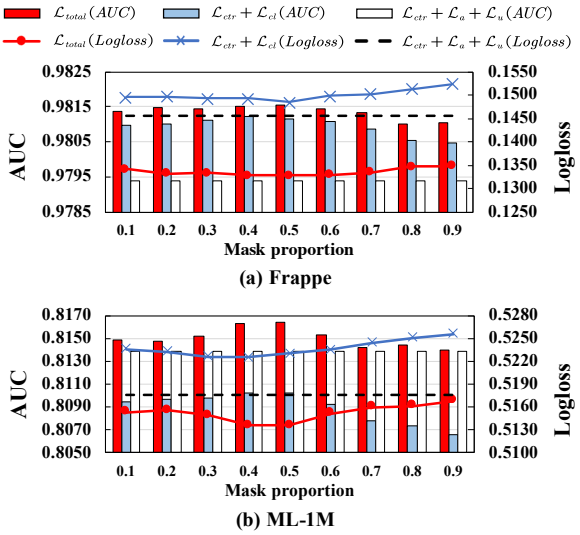


Figure 5: Impact of random mask proportion.

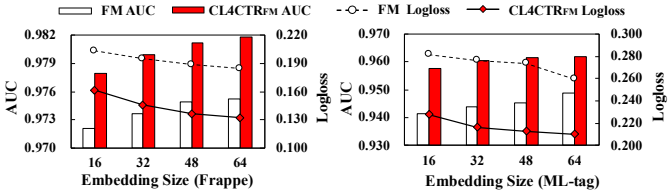


Figure 6: Impact of embedding size on FM and $CL4CTR_{FM}$.

4.5.3 Embedding Size. We change the embedding size from 16 to 64 with a step of 16 in the embedding layer and show the experimental results in Figure 6. It can be observed that the performance of $CL4CTR_{FM}$ is improved substantially with the embedding sizes increasing. Meanwhile, CL4CTR can improve the performance of FM with all embedding sizes. Furthermore, compared with the embedding size of 64 in FM, $CL4CTR_{FM}$ achieves better performance with a small size of 16. This means we can reduce the parameters while achieving better results by applying CL4CTR on FM.

5 CONCLUSION

In this paper, we propose a novel framework named Contrastive Learning for CTR prediction (CL4CTR), which directly improves the quality of feature representations, especially for low frequency features. In CL4CTR, we introduce a contrastive module to improve the quality and generalizability of the feature representations by fully utilizing the self-supervised signals from the features. Furthermore, considering the unique characteristics of CTR prediction tasks, we propose two constraints in contrastive learning: feature alignment and feature uniformity, which are used to regularize feature representations. The extensive experimental results demonstrate the excellent effectiveness and compatibility of our proposed CL4CTR on four public datasets.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 62172106 and 61932007.

REFERENCES

[1] 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*. 3757–3766.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[3] Zekai Chen, Fangtian Zhong, Zhumin Chen, Xiao Zhang, Robert Pless, and Xiuzhen Cheng. 2021. DCAP: Deep Cross Attentional Product Network for User Response Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 221–230.

[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] Weiyu Cheng, Yanyan Shen, and Linpeng Huang. 2020. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3609–3616.

[6] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 6894–6910.

[7] 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*.

[8] Wei Guo, Can Zhang, Zhicheng He, Jiarui Qin, Huifeng Guo, Bo Chen, Ruiming Tang, Xiuqiang He, and Rui Zhang. 2022. Miss: Multi-interest self-supervised learning framework for click-through rate prediction. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 727–740.

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[10] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. 355–364.

[11] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[12] Tongwen Huang, Qingyun She, Zhiqiang Wang, and Junlin Zhang. 2020. GateNet: Gating-Enhanced Deep Network for Click-Through Rate Prediction. *arXiv preprint arXiv:2007.03519* (2020).

[13] 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*. 169–177.

[14] Yuchin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware factorization machines for CTR prediction. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 43–50.

[15] Dongha Lee, SeongKu Kang, Hyunjun Ju, Chanyoung Park, and Hwanjo Yu. 2021. Bootstrapping user and item representations for one-class collaborative filtering. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 317–326.

[16] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. 2020. Interpretable click-through rate prediction through hierarchical attention. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 313–321.

[17] 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*. 1754–1763.

[18] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature generation by convolutional neural network for click-through rate prediction. In *The World Wide Web Conference*. 1119–1129.

[19] Wantong Lu, Yantao Yu, Yongzhe Chang, Zhen Wang, Chenhui Li, and Bo Yuan. 2020. A Dual Input-aware Factorization Machine for CTR Prediction. In *IJCAI*. 3139–3145.

[20] Yuanfei Luo, Hao Zhou, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. 2020. Network on network for tabular data classification in real-world applications. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2317–2326.

[21] 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 *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1298–1307.

[22] Junwei Pan, Jian Xu, Alfonso Lobos Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted factorization machines for click-through rate prediction in display advertising. In *Proceedings of the 2018 World Wide Web Conference*. 1349–1357.

[23] Yujie Pan, Jiangchao Yao, Bo Han, Kunyang Jia, Ya Zhang, and Hongxia Yang. 2021. Click-through Rate Prediction with Auto-Quantized Contrastive Learning. *arXiv preprint arXiv:2109.13921* (2021).

[24] Jiarui Qin, Weinan Zhang, Xin Wu, Jiarui Jin, Yuchen Fang, and Yong Yu. 2020. User behavior retrieval for click-through rate prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2347–2356.

[25] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2018. Product-based neural networks for user response prediction over multi-field categorical data. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 1–35.

[26] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 1–22.

[27] Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*. 521–530.

[28] 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*. 1161–1170.

[29] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. Fm2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the Web Conference 2021*. 2828–2837.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[31] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. 2021. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*. PMLR, 10530–10541.

[32] Feng Wang and Huaping Liu. 2021. Understanding the behaviour of contrastive loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2495–2504.

[33] 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 *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[34] 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.

[35] Ruoxi Wang, Rakesh Shivanna, Derek Z Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H Chi. 2020. DCN-M: Improved Deep & Cross Network for Feature Cross Learning in Web-scale Learning to Rank Systems. *arXiv preprint arXiv:2008.13535* (2020).

[36] Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*. PMLR, 9929–9939.

[37] Zhiqiang Wang, Qingyun She, and Junlin Zhang. 2021. MaskNet: introducing feature-wise multiplication to CTR ranking models by instance-guided mask. *arXiv preprint arXiv:2102.07619* (2021).

[38] Shu Wu, Feng Yu, Xueli Yu, Qiang Liu, Liang Wang, Tieniu Tan, Jie Shao, and Fan Huang. 2020. TFNet: Multi-Semantic Feature Interaction for CTR Prediction. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1885–1888.

[39] 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 *IJCAI*.

[40] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE.

[41] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, and Zi Huang. 2022. Self-Supervised Learning for Recommender Systems: A Survey. *arXiv preprint arXiv:2203.15876* (2022).

[42] Yantao Yu, Zhen Wang, and Bo Yuan. 2019. An Input-aware Factorization Machine for Sparse Prediction. In *IJCAI*. 1466–1472.

[43] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. 2021. Deep learning for click-through rate estimation. In *IJCAI*.

[44] Zihao Zhao, Zhiwei Fang, Yong Li, Changping Peng, Yongjun Bao, and Weipeng Yan. 2020. Dimension Relation Modeling for Click-Through Rate Prediction. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2333–2336.

[45] Zhishan Zhao, Sen Yang, Guohui Liu, Dawei Feng, and Kele Xu. 2022. FINT: Field-Aware Interaction Neural Network for Click-Through Rate Prediction. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3913–3917.

[46] Xin Zhou, Aixin Sun, Yong Liu, Jie Zhang, and Chunyan Miao. 2021. SelfCF: A Simple Framework for Self-supervised Collaborative Filtering. *arXiv preprint arXiv:2107.03019* (2021).