MCRF: Enhancing CTR Prediction Models via Multi-channel Feature Refinement Framework

Fangye Wang[1,2], Hansu Gu[3], Dongsheng Li[4], Tun Lu[1,2], Peng Zhang[1,2](✉), and Ning Gu[1,2]

[1] School of Computer Science, Fudan University, Shanghai, China
{fywang18,lutun,zhangpeng_,ninggu}@fudan.edu.cn
[2] Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China
[3] Seattle, USA
[4] Microsoft Research Asia, Shanghai, China
dongsli@microsoft.com

Abstract. Deep learning methods have recently achieved huge success in Click-Through Rate (CTR) prediction tasks mainly due to the ability to model arbitrary-order feature interactions. However, most of the existing methods directly perform feature interaction operations on top of raw representation without considering whether raw feature representation only obtained by embedding layer is sufficiently accurate or not, which leads to suboptimal performance. To address this issue, we design a model-agnostic and lightweight structure named Gated Feature Refinement Layer (GFRL), which dynamically generates flexible and informative feature representation by absorbing both intra-field and contextual information based on specific input instances. We further propose a generalized framework Multi-Channel Feature Refinement Framework (MCRF), which utilizes GFRLs to generate multiple groups of embedding for richer interactions without exploding the parameter space. GFRL and MCRF can be well generalized in many existing CTR prediction methods to improve their performance. Extensive experiments on four real-world datasets show that GFRL and MCRF can achieve statistically significant performance improvements when applied to mainstream CTR prediction algorithms. The integration of MCRF and FM can further push the state-of-the-art performance of CTR prediction tasks. We also identify that refining feature representation is another fundamental but vital direction to boost the performance of CTR prediction models.

Keywords: CTR prediction · Feature refinement · Recommendation

1 Introduction

CTR prediction, which aims to estimate the likelihood of an ad or item being clicked, is essential to the success of several applications such as computational advertising [13], and recommender systems [1]. Accurate CTR prediction plays a vital role to improve revenue, and customer engagement of the platform [4,12]. However, in CTR tasks, the input data is typically high-dimensional and sparse multi-field categorical features, which takes great difficulty for making an accurate prediction. Many models have been proposed such as Logistic Regression (LR) [17], and Factorization Machines (FM) based methods [5,10,16,19,23].

With its strong ability to learn high-order feature interaction information, deep learning-based models gain popularity and achieve great success in CTR prediction tasks [1,2,4,5,25]. Meanwhile, various researchers improve performance by modeling sophisticated high-order feature interaction, including DCN [20], DeepFM [4], xDeepFM [11], FiBiNET [9], AutoInt [18], TFNet [21], etc. Most CTR models comprise of three components: 1) embedding layer; 2) feature interaction layer; 3) prediction layer. However, despite the great success of the above-mentioned methods, two key research challenges remain unsolved.

Firstly, obtaining more informative and flexible feature representation before performing feature interaction remains challenging. Precise feature representation is the basis of feature interaction and prediction layer. However, raw feature representation only obtained by embedding layer may not be sufficiently accurate. NON utilizes intra-field information to refine raw features [13]. However, for the same feature in different input instances, its representation is still fixed in NON. In the NLP area, contextual information is leveraged in Bert [3] and other methods to obtain word embedding dynamically based on the sentence context where the word appears. It is also reasonable to refine feature representation according to other concurrent features in each input instance for CTR prediction tasks. For example, we have two input instances: {*female, student, red, lipstick*} and {*female, student, white, computer*}. Apparently, the feature *"female"* is more important for click probability in the former instance, which is affected by the instance it appears [24]. So it remains a big challenge to learn more flexible feature representation for downstream feature interaction and prediction layer.

Secondly, given the huge possible combinations of raw features, interactions are usually numerous and sparse. FM [16] and most deep methods utilize a single embedding for each feature but fail to capture multiple aspects of the same feature. Field-aware factorization machine (FFM) introduces a separate feature embedding for each field [10]. However, it sometimes creates more parameters than necessary. Meanwhile, most parallel-structured models (i.e., DeepFM [4], DCN [20]) perform different sub-networks to capturing feature interaction based on the unique shared embedding. Intuitively, different interaction functions should contain discriminative embedding for optimal performance. Those bring up the question of how to generate multiple groups of embeddings for richer feature interaction information without exploding the parameter space.

To address the above two issues, we firstly propose a model-agnostic and lightweight structure named **G**ated **F**eature **R**efinement **L**ayer (**GFRL**), which dynamically refines raw feature's embedding by absorbing both intra-field and contextual information. Based on GFRL, we further propose a generalized **M**ulti-**C**hannel Feature **R**efinement **F**ramework (**MCRF**), which is comprised of three layers. In multi-channel refinement layer, we employ several independent GFRLs to generate multiple refined embeddings for further feature interactions. In aggregation layer, we adopt mainstream feature interaction operations in each channel and aggregate all generated interactions using attention mechanism. Finally, we utilize a DNN to generate the final output in prediction layer.

In summary, our main contributions are as follows: (1) We design a lightweight and model-agnostic module GFRL to refine feature representation by integrating intra-field and contextual information simultaneously. (2) Based on GFRL, we propose MCRF, which enables richer feature interaction information by generating multiple discriminative refined embedding in different channels. (3) Comprehensive experiments are conducted to demonstrate the effectiveness and compatibility of GFRL and MCRF on four datasets.

2 Related Work

FM-based methods are widely used to address CTR prediction tasks. FM [16] models all pairwise interactions using factorized parameters, which can handle large-scale sparse data via dimension reduction. Because of its efficiency and robustness, there are many subsequent works to improve FM from different angles, such as FFM [10], Field-weighted Factorization Machines (FwFM) [14], Attentional FM (AFM) [22], Neural FM (NFM) [5], Input-aware FM (IFM) [24], etc.

In recent years, leveraging deep learning to model complex feature interaction draws more interest. Models, such as FNN [25], WDL [1], DeepFM [4] utilizes plain DNN module to learn higher-order feature interactions. Meanwhile, DeepFM replaces the wide component of WDL with FM to reduce expensive manual feature engineering. xDeepFM [11] improve DeepFM by a novel Compressed Interaction Network (CIN) which learns high-order feature interactions explicitly. PNN [15] feeds the concatenation of embedding and pairwise interactions into DNN. DCN [20] model feature interactions of bounded-degree by using a cross network. And self-attention is a commonly used network to capture high-order interaction, such as AutoInt [18].

All aforementioned CTR prediction methods feed the raw embeddings to feature interaction layer for modeling arbitrary-order feature interaction. Differ to model feature interactions, researchers start to focus on refining feature representation before the feature interactions. FiBiNET [9] utilize the Squeeze-Excitation Network (SENET) [8] to generate SENET-like embedding for further feature interactions by assigning each feature unique importance. Similarly, IFM [24] also improves FM by learning feature importance with the proposed Factor Estimating Network (FEN). NON [13] leverages field-wise network to extract intra-field information and enrich raw features. In our work, GFRL considers intra-field information and contextual information simultaneously and refines original embedding more flexible and fine-grained.

3 The Structure of MCRF

We aim to refine raw features by encoding both intra-field information and contextual information and enable multiple embedding channels for richer useful

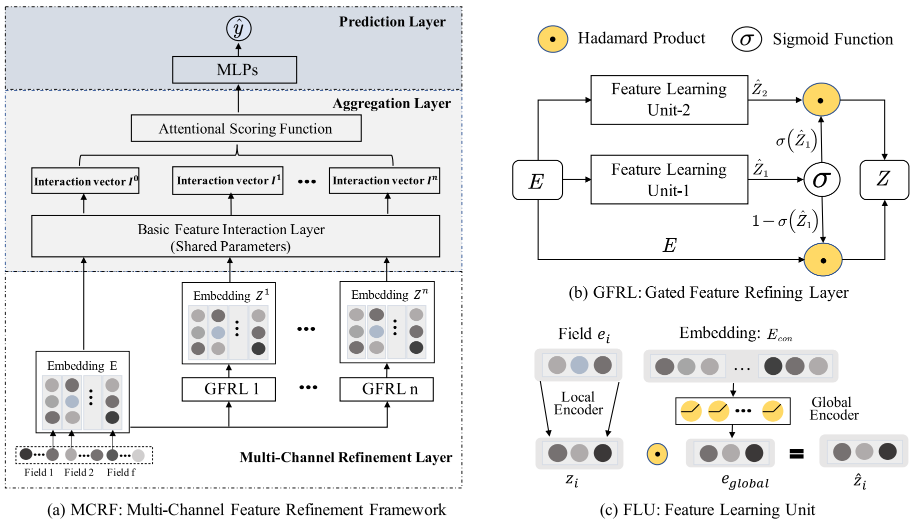


Fig. 1. Architectures of the proposed (a) MCRF, (b) GFRL, and (c) FLU.

feature interactions. To this end, we propose GFRL and MCRF for CTR prediction. As depicted in Fig. 1a, MCRF is composed of three layers:

- **Multi-Channel Refinement Layer**. In multi-channel refinement layer, we propose a model-agnostic structure named GFRL, which is applied on top of Embedding Layer to refine original embedding. GFRL contains Feature Learning Unit (FLU), and Gate Unit. FLU focuses on integrating both intra-field and contextual information and Gate Unit controls the selection probability of the supplementary embedding and original embedding for generating final refined representation.
- **Aggregation Layer**. In aggregation layer, we utilize attention mechanism to select and aggregate useful feature interactions based on multiple refined embeddings enabled by the multi-channel design.
- **Prediction Layer**. In prediction layer, a DNN is used to output the final prediction based on the features from the aggregation layer.

3.1 Embedding Layer

In most CTR prediction tasks, the raw data is typically in a multi-field categorical format, which is usually represented as a high-dimensional sparse vector (binary) by one-hot encoding [4,12]. For example, an input instance (Gender = Female, Item = Lipstick, Color = Red, Name = Amy) can be represented as:

$$\underbrace{(0,1)}_{\text{Gender=Female}} \quad \underbrace{(0,\ldots,1,0,0)}_{\text{Item=Lipstick}} \underbrace{(0,1,\ldots,0,0)}_{\text{Color=Red}} \underbrace{(1,0,0,\ldots,0)}_{\text{Name=Amy}} . \tag{1}$$

In most DNN-based methods, an embedding layer is used to transform high-dimensional sparse feature data into low-dimensional dense vectors. Each input instance has f feature fields, after inputting into the embedding layer, each field i $(1 \leq i \leq f)$ is represented by low-dimensional vector $e_i \in \mathbb{R}^d$. Each input instance can be represented by embedding matrix $E = [e_1, ..., e_i, ..., e_f] \in \mathbb{R}^{f \times d}$, where d is the dimension of embedding layer, e_i is the embedding of field i.

3.2 Gated Feature Refinement Layer

GFRL dynamically refines raw feature embedding by absorbing intra-field and contextual information simultaneously. And more informative and expressive representation can boost the performance of other basic models. As shown in Fig. 1b, GFRL consists of **FLU** and **Gate Unit** presented as follows.

Feature Learning Unit. FLU focuses on encoding intra-field and contextual information explicitly. We use two independent FLUs to generate supplementary features and a group of weight matrix. As illustrated in Fig. 1c, FLU is comprised of two steps: encoder step and combination step.

Encoder. This step encodes intra-field and contextual information by a local and global encoder. In local encoder, we assign a separate Fully Connected (FC) layer for each field. The parameters in FCs are used to store the intra-field information, then the output is computed by:

$$z_i = FC_i(e_i), \tag{2}$$

where e_i, z_i and FC_i are the original embedding, output embedding, and FC of the i-th field respectively. In practice, each field has the same structure as the FC, so we calculate them all together by applying matrix multiplication once (stacking inputs and weights of each FC [13]). We denote $B_i \in \mathbb{R}^{b \times d}$ and $W_i \in \mathbb{R}^{d \times d}$ as the batch of inputs and weights of the FC_i, where b is the mini-batch size. Formally, we formulate the local encoder as follows:

$$Z_{ori} = stack[B_1, ..., B_i, ..., B_f] \in \mathbb{R}^{f \times b \times d}, \tag{3}$$

$$W_{local} = stack([W_1, ..., W_i, ..., W_f]) \in \mathbb{R}^{f \times d \times d}, \tag{4}$$

$$Z_{local} = matmul(Z_{ori}, W_{local}) + b_l \in \mathbb{R}^{f \times b \times d}, \tag{5}$$

where d is the input and output dimension of FC_i. $b_l \in \mathbb{R}^{f \times 1 \times d}$ is bias term and $matmul$ is batch matrix multiplication which is supported by Pytorch.[1] $Z_{local} = [z_1, ..., z_i, ..., z_f]$ is the output of local encoder.

After encoding intra-field information, we proceed to capture contextual information with a global encoder. Contextual information indicates the environment in which a feature is located. The concatenate all the embedding is represented by $E_{con} \in \mathbb{R}^{fd}$, and feed them to a simple DNN as follows:

$$e_{global} = \sigma_2(W_2\sigma_1(W_1E_{con} + b_1) + b_2), \tag{6}$$

[1] https://pytorch.org/docs/1.1.0/torch.html?#torch.matmul.

where $W_1 \in \mathbb{R}^{d_k \times fd}$, $W_2 \in \mathbb{R}^{d \times d_k}$, b_1 and b_2 are learnable parameters; $\sigma(\cdot)$ is the ReLU function. We model e_{global} as the contextual information, which represents the specific context information of each instance.

Combination. In this step, we combine intra-field and contextual information, using the following equation:

$$\hat{z}_i = F(z_i, e_{global}) = z_i \odot e_{global} \in \mathbb{R}^d, \tag{7}$$

where $F(\cdot)$ is a function, we choose the Hadamard product denoted by $\odot$ here, which is parameter-free and widely used in recommendation systems [4,15,16]. Finally, the supplementary features are generated as $\hat{Z} = [\hat{z}_1, ..., \hat{z}_i, ..., \hat{z}_f] \in \mathbb{R}^{f \times d}$, which explicitly integrates both intra-field and contextual information.

Compared to NON, FLU further absorbs contextual information, enabling the same feature to obtain various feature representations dynamically. It is also affected by the specific instance it appears, represented by contextual information here. To further enhance the expressive power of feature learning, we design a Gate Unit to select salient features from the original and supplementary embedding adaptively.

Gate Unit. Inspired by Long Short Term Memory Network (LSTM) [7], we leverage gate mechanism to obtain refined features by controlling the selection probability of supplementary features and raw features. As shown in Fig. 1b, we compute one supplementary embedding matrix $\hat{Z}_2$ and one group of weight matrix $\hat{Z}_1$ by two independent FLUs. Then we generate a group of probabilities $\boldsymbol{\sigma}$ by adding a sigmoid function after matrix $\hat{Z}_1$, which can control information flow in bit-level. Finally, we obtain the refined feature embedding Z as follows:

$$\hat{Z}_1 = FLU_1(E) \in \mathbb{R}^{f \times d}, \tag{8}$$

$$\hat{Z}_2 = FLU_2(E) \in \mathbb{R}^{f \times d}, \tag{9}$$

$$\boldsymbol{\sigma} = sigmoid(\hat{Z}_1) \in \mathbb{R}^{f \times d}, \tag{10}$$

$$Z = \hat{Z}_2 \odot \boldsymbol{\sigma} + E \odot (1 - \boldsymbol{\sigma}), \tag{11}$$

where $Z \in \mathbb{R}^{f \times d}$ denotes the refined features. $\boldsymbol{\sigma}$ is the weight matrix, which controls the information flow like a gate in bit-level. $\hat{Z}_2 \odot \boldsymbol{\sigma}$ represents the portion of each element that can be retained in supplementary features. $E \odot (1 - \boldsymbol{\sigma})$ denotes a selective "forgetness" of the raw features, and it only retains the important portion of the raw features. These two parts are complementary to each other. It should be noted that we do not use activation function in *local encoder*. As we have applied ReLU in Eq. 6. If functions like ReLU were used in *local encoder* again, the outputs of FLU would be greater than 0, which would cause all the probabilities to be greater than 0.5. In that case, the gate function would not help identify useful features. We believe that the use of gate unit helps improve the stability of the GFRL module.

In summary, GFRL utilizes two FLUs to generate supplementary features and weights matrix by integrating intra-field and contextual information and

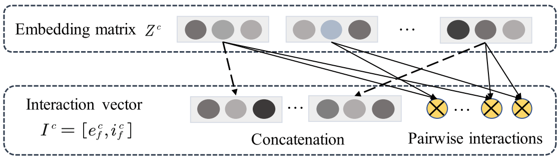


Fig. 2. The interaction layer of IPNN [15].

leverages gate unit to choose important information from both the raw and the supplementary features. It can be used as a building block in a plug-and-play fashion to improve base models' performance, we will discuss this in Sect. 4.3.

3.3 Multi-channel Feature Refinement Framework

To enable multiple embedding channels for richer feature interactions, we design MCRF (Fig. 1a) based on GFRL. It is comprised of three layers: Multi-Channel Refinement Layer, Aggregation Layer, and Prediction Layer.

Multi-channel Refinement Layer. Given the numerous combinations of raw features, feature interactions usually suffer from sparsity issues. FM-based and most deep methods address the issue by proposing feature embeddings, improving the model's generalization performance but performing feature interaction only based on the raw embedding matrix. FFM learns a separate feature embedding for each field, although achieving a significant performance boost, cause an explosion of parameters and compromise memory usage. We utilize multiple GFRLs to output multiple refined feature matrices that only scale the parameter space based on the number of parameters within GFRL and then perform feature interaction independently. In this way, we leverage richer embeddings through multiple channels without exploding the parameter space. We input the raw embedding matrix E into multiple GFRLs as follows:

$$Z^c = GFRL(E) \in \mathbb{R}^{f \times d}, \tag{12}$$

where Z^c denotes refined embedding in c-th $(1 \leq c \leq n)$ channel. We denote $\mathbb{Z} = (E, Z^1, ..., Z^c, ..., Z^n)$ as all the embedding matrices, and we also keep the raw embedding E. When channel number is zero, we directly use the base model.

Aggregation Layer. In the previous layer, we obtain $n+1$ embedding matrices, n is the channel numbers. In this layer, we first apply basic feature interaction operation in each channel to get a feature vector. Then all vectors are aggregated by the attention mechanism. To be mentioned, any other advanced interaction operations can be applied to generate feature vector, e.g., bi-interaction in NFM [5], attention in AFM [22], etc. For ease of presentation, we take the interaction operation in IPNN as an example, as shown in Fig. 2.

IPNN concatenates pairwise inner-product and input vectors to model feature interaction. We denote $\boldsymbol{e}_f^c$ and $\boldsymbol{i}_f^c$ as the embedded features and the inner-product features in the c-th channel. We can calculate $\boldsymbol{e}_f^c$ and $\boldsymbol{i}_f^c$ as follow:

$$\boldsymbol{e}_f^c = concat(Z^c) = concat([z_1^c; ...; z_i^c; ...; z_f^c]), \tag{13}$$

$$\boldsymbol{i}_f^c = [p_{(1,2)}, p_{(2,3)}, ..., p_{(f-1,f)}], \tag{14}$$

$$p_{(i,j)} =< z_i^c, z_j^c >, \tag{15}$$

where $Z^c \in \mathbb{R}^{f \times d}$ and z_i^c are the embedding matrix and i-th field embedding in c-th channel respectively. $p_{(i,j)}$ is the value of the "inner-product" operation between the i-th and j-th features. Concatenating $\boldsymbol{e}_f^c$ and $\boldsymbol{i}_f^c$ obtains the interaction vector $I^c = [\boldsymbol{e}_f^c, \boldsymbol{i}_f^c] \in \mathbb{R}^{n_f}$ in c-th channel, where $n_f = fd + \frac{(f-1)f}{2}$. Considering the input $\mathbb{Z}$, we obtain a total of $n+1$ interaction vectors $\mathbb{I} = [I^0, I^1, ..., I^c, ..., I^n] \in \mathbb{R}^{(n+1) \times n_f}$.

As the embedding matrix Z^c is different in each channel, each interaction vector captures different interaction information. To aggregate all interaction vectors, we adopt attention on vector set $\mathbb{I}$. Formally, the attentional aggregation representation $\mathbb{I}_{agg}$ can be described as follows:

$$\mathbb{I}_{agg} = AttentionAgg(\mathbb{I}) = \sum_{c=0}^{n} \alpha^c I^c, \tag{16}$$

$$\alpha^c = \frac{\exp\left(\boldsymbol{h}^T \operatorname{ReLU}\left(W^c I^c\right)\right)}{\sum_{c' \in n+1} \exp\left(\boldsymbol{h}^T \operatorname{ReLU}\left(W^c I^{c'}\right)\right)}, \tag{17}$$

where α^c denotes the attention score in c-th channel; $W^c \in \mathbb{R}^{s \times n_f}$ are learnable parameters; $\boldsymbol{h} \in \mathbb{R}^s$ is the context vector; and s is the attention size. Experiments show attention size does not affect results apparently, we set s equals to n_f.

Briefly, we implement basic interaction operation in each channel for generating the interaction vector, and each vector stores supplementary interaction information. For simplicity, we share the learning parameters in basic feature interaction layer. Actually, most operations like inner product, outer product and bi-interaction are parameter-free. Finally, we utilize an attention mechanism to aggregate all the interaction vectors generated in multiple channels.

Prediction Layer. After the aggregation layer, we collect the aggregated representation $\mathbb{I}_{agg}$. Then, we feed it into a DNN for the final prediction $\hat{y}$ as follows:

$$\hat{y} = \operatorname{sigmoid}\left(\operatorname{DNN}\left(\mathbb{I}_{agg}\right)\right). \tag{18}$$

The learning process aims to minimize the following objective function:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log\left(\hat{y}_i\right) + (1 - y_i) * \log\left(1 - \hat{y}_i\right)\right) \tag{19}$$

where $y_i \in \{0, 1\}$ is the true label and $\hat{y_i} \in (0, 1)$ is the predicted CTR, and N is the total size of samples.

4 Experiments

To comprehensively evaluate MCRF and its major component GFRL, we conduct experiments by answering five crucial research questions. During the process, we essentially break down and evaluate different components of MCRF, so the whole experiment serves as a comprehensive ablation study.

- (Q1) How do GFRL and MCRF perform compared to mainstream methods while applying them to FM?
- (Q2) Can GFRL and MCRF improve the performance of other state-of-the-art CTR prediction methods?
- (Q3) What are the important components (FLU and Gate Unit) in GFRL?
- (Q4) How does GFRL perform compared to other structures?
- (Q5) How does the number of channels impact the performance of MCRF?

4.1 Experimental Setup

Datasets. We conduct empirical experiments on four real-world datasets. Table 1 lists the statistics of four real-world datasets.

Criteo[2] is a famous benchmark dataset, which contains 13 continuous feature fields and 26 categorical feature fields. It includes 45 million user click records. We use the last 5 million sequential records for test and the rest for training and validation [6]. We further remove the infrequent feature categories and treat them as a "$\langle unknown \rangle$" category, where the threshold is set to 10. And numerical features are discretized by the function $discrete(x) = \lfloor log^2(x) \rfloor$, where $\lfloor . \rfloor$ is the floor function, which is proposed by the winner of Criteo Competition.[3]

Avazu[4] is provided by Avazu to predict whether a mobile ad will be clicked. We also remove the infrequent feature categories, the threshold is set to five [18].

Frappe[5] is used for context-aware recommendation. Its target value indicates whether the user has used the app [2].

MovieLens[6] is used for personalized tag recommendation. Its target value denotes whether the user has assigned a particular tag to the movie [2]. We strictly follow AFM [22] and NFM [5] to divide the training, validation and test set for MovieLens and Frappe.

Evaluation Metrics. We use **AUC** (Area Under the ROC curve) and **Logloss** (cross-entropy) as the metrics. Higher AUC and lower Logloss indicate better performance [2,12,13]. It should be noted that **0.001**-level improvement in AUC or Logloss is considered significant [2,13,20] and likely to lead to a significant increase in online CTR prediction [4,12]. Considering the huge daily turnover of platforms, even a few lifts in CTR bring extra millions of dollars each year [12].

[2] https://www.kaggle.com/c/criteo-display-ad-challenge.
[3] https://www.csie.ntu.edu.tw/~r01922136/kaggle-2014-criteo.pdf.
[4] https://www.kaggle.com/c/avazu-ctr-prediction.
[5] https://www.baltrunas.info/context-aware/frappe.
[6] https://grouplens.org/datasets/movielens/.

Table 1. Datasets statistics (K indicates thousand.)

Datasets	Positive	Training	Validation	Test	Features	Fields
Criteo	26%	35,840K	5,000K	5,000K	1,087K	39
Avazu	17%	32,343K	4,043K	4,043K	1,544K	23
Frappe	33%	202K	58K	29K	5K	10
MovieLens	33%	1,404K	401K	201K	90K	3

Comparison Methods. We select the following existing CTR prediction models for comparison, including FM [16], IFM [24], FFM [10], FwFM [14], WDL [1], NFM [5], DeepFM [4], xDeepFM [11], IPNN [15], OPNN [15], FiBiNET [9], AutoInt+ [18], AFN+ [2], TFNet [21], NON [13]. As xDeepFM, FiBiNET and AFN has outperformed LR, FNN, AFM, DCN, we do not present their results. Meanwhile, our experiments also show the same conclusions.

Reproducibility. All methods are optimized by Adam optimizer with a learning rate of 0.001, and a mini-batch size of 4096. To avoid overfitting, early-stopping is performed according to the AUC on the validation set. We fix the dimension of field embedding for all models to be a 10 for Criteo and Avazu, 20 for Frappe and MovieLens, respectively. Borrowed from previous work [2,4,9], for models with the DNN, the depth of layer is set to 3, and the number of neurons per layer is 400. ReLU activation functions are used, and the dropout rate is set to 0.5. For other models, we fetch the best setting from its original literature.

To ensure fair comparison, for each model on each dataset, we run the experiments five times and report the average value. Notably, all the standard deviations are in the order of $\mathbf{1e{-}4}$, showing our results are very stable. Furthermore, the two-tailed pairwise t-test is performed to detect significant differences between our proposed methods and the best baseline methods.

4.2 Overall Performance Comparison (Q1)

To verify the effectiveness of GFRL and MCRF, we choose FM as the basic method, represented by FM+GFRL and FM+MCRF-4.[7] FM is the most simple and effective feature interaction method. The performance of different models on the test set is summarized in Table 2. The key observations are as follow:

Firstly, most of the deep-learning models outperform shallow models. One of the reasons is that many deep models apply well-designed feature interaction operations, such as CIN in xDeepFM, self-attention in AutoInt, etc.

Secondly, by applying GFRL on top of embedding layer, FM+GFRL outperforms FM and other mainstream models, only second to FM+MCRF-4. It

[7] As the output of FM is a scalar value, we omit the attention layer when we apply MCRF with FM. Meanwhile, we feed all outputs in multiple channels to a simple LR than DNN in the prediction layer, which makes FM+MCRF more efficient.

Table 2. Overall performance comparison. ($\star : p < 10^{-2}$; $\star\star : p < 10^{-4}$)

Datasets		Criteo		Avazu		Frappe		MovieLens	
Model		AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
Shallow models	FM	0.8028	0.4514	0.7812	0.3796	0.9708	0.1934	0.9391	0.2856
	FFM	0.8066	0.4477	0.7873	0.3758	0.9780	0.1980	0.9485	0.2682
	FwFM	0.8072	0.4470	0.7855	0.3770	0.9740	0.2164	0.9457	0.2840
	IFM	0.8066	0.4470	0.7839	0.3778	0.9765	0.1896	0.9471	0.2853
Deep models	WDL	0.8068	0.4474	0.7880	0.3753	0.9776	0.1895	0.9403	0.3045
	NFM	0.8057	0.4483	0.7828	0.3782	0.9746	0.1915	0.9437	0.2945
	DeepFM	0.8084	0.4458	0.7875	0.3758	0.9789	0.1770	0.9465	0.3079
	xDeepFM	0.8086	0.4456	0.7884	0.3751	0.9792	0.1889	0.9480	0.2889
	IPNN	0.8088	0.4454	0.7882	0.3754	0.9791	0.1759	0.9490	0.2785
	OPNN	0.8096	0.4446	0.7885	0.3752	0.9795	0.1805	0.9497	0.2704
	FiBiNET	0.8089	0.4453	0.7887	0.3746	0.9756	0.2767	0.9435	0.3427
	AutoInt+	0.8088	0.4456	0.7882	0.3751	0.9786	0.1890	0.9501	0.2813
	AFN+	0.8095	0.4447	0.7886	0.3747	0.9791	0.1824	0.9509	0.2583
	TFNet	0.8092	0.4449	0.7885	0.3745	0.9787	0.1942	0.9493	0.2714
	NON	0.8095	0.4446	0.7886	0.3748	0.9792	0.1813	0.9505	0.2625
Our models	FM+GFRL	0.8107**	0.4439**	0.7889**	0.3746**	0.9803*	0.1774*	0.9549*	0.2453*
	FM+MCRF-4	**0.8114****	**0.4433****	**0.7898****	**0.3740****	**0.9821***	**0.1720***	**0.9589****	**0.2409****

demonstrates that the refined feature representation is extremely efficient, which captures more information by absorbing intra-field and contextual information. And compared to modeling feature interaction, refining feature representation is more effective, considering FM only models second-order feature interaction.

Finally, with the help of four groups of refined embeddings, FM+MCRF-4 achieves the best performance. Specifically, it significantly outperforms basic FM 1.07%, 1.10%, 1.16% and 2.11% in terms of AUC(1.79%, 1.48%, 11.07%, and 15.65% in terms of Logloss) on four datasets. Those facts verify that the additional group of embeddings can provide more information. In subsection 4.6, we will discuss the impact of channel numbers in detail.

4.3 Compatibility of GFRL and MCRF with Different Models (Q2)

We implement GFRL and MCRF on five base models: FM, AFM, NFM, IPNN, and OPNN. GFRL can be applied after embedding layer to generate informative and expressive embedding. For a fair comparison, we only use one channel in MCRF to obtain another group feature embedding. The results are presented in Table 3. We focus on BASE, GFRL, and MCRF-1 and draw the following conclusions.

All base models achieve better performance after applying GFRL and MCRF, demonstrating the effectiveness and compatibility of GFRL and MCRF. Meanwhile, MCRF-1 consistently outperforms the building block GFRL, which verifies that MCRF-1 successfully aggregates more useful interactions from raw features and refined features by attention mechanism.

Compared to IPNN and OPNN, FM-based models get better improvements by applying GFRL and MCRF-1. Especially, FM+GFRL and FM+MCRF-1

Table 3. Model performance of different modules based upon various basic mainstream models on Criteo and Avazu datasets.

(a) Criteo Dataset

Models	FM		AFM		NFM		IPNN		OPNN	
Modules	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
BASE	0.8028	0.4514	0.7999	0.4535	0.8057	0.4483	0.8088	0.4454	0.8096	0.4444
SENET	0.8057	0.4484	0.7990	0.4544	0.8052	0.4486	0.8090	0.4451	0.8093	0.4449
FEN	0.8066	0.4470	0.7954	0.4570	0.8063	0.4475	0.8092	0.4451	0.8088	0.4454
FWN	0.8049	0.4491	0.7884	0.4637	0.8070	0.4476	0.8090	0.4451	0.8098	0.4446
FLU	0.8098	0.4446	0.8082	0.4461	0.8097	0.4445	0.8093	0.4450	0.8100	0.4445
GFRL	<u>0.8107</u>	<u>0.4439</u>	<u>0.8091</u>	<u>0.4453</u>	<u>0.8103</u>	<u>0.4441</u>	<u>0.8097</u>	<u>0.4444</u>	<u>0.8104</u>	<u>0.4440</u>
MCRF-1	**0.8110**	**0.4435**	**0.8092**	**0.4452**	**0.8106**	**0.4437**	**0.8099**	**0.4444**	**0.8107**	**0.4439**

(b) Avazu dataset.

Models	FM		AFM		NFM		IPNN		OPNN	
Modules	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss	AUC	Logloss
BASE	0.7812	0.3796	0.7727	0.3849	0.7828	0.3782	0.7882	0.3754	0.7885	0.3752
SENET	0.7831	0.3785	0.7743	0.3847	0.7847	0.3743	0.7883	0.3751	0.7887	0.3747
FEN	0.7839	0.3778	0.7731	0.3854	0.7830	0.3783	0.7886	0.3746	0.7885	0.3751
FWN	0.7846	0.3776	0.7731	0.3854	0.7853	0.3768	0.7884	0.3750	0.7886	0.3748
FLU	0.7875	0.3755	0.7840	0.3780	0.7881	0.3750	0.7885	0.3750	0.7888	0.3746
GFRL	<u>0.7889</u>	<u>0.3746</u>	<u>0.7854</u>	<u>0.3773</u>	<u>0.7883</u>	<u>0.3750</u>	<u>0.7887</u>	<u>0.3746</u>	<u>0.7890</u>	<u>0.3745</u>
MCRF-1	**0.7891**	**0.3745**	**0.7856**	**0.3768**	**0.7887**	**0.3747**	**0.7889**	**0.3745**	**0.7895**	**0.3742**

achieve the best result on Criteo dataset, although the other four basic models have more complex structures. Those facts show that inaccurate feature representation limits the effectiveness of these models, and learning precise and expressive representation is more basic but practical.

The key structure GFRL can improve the performance of base models and accelerate the convergence. Figure 3 exhibits the test AUC of three models during the training process before and after applying GFRL. Base models with GFRL can speed up the training process significantly. Specifically, within five epochs, base models with GFRL outperform the best results of the base models, indicating that refined representation generated by GFRL can capture better features and optimize the subsequent operations, and lead to more accurate prediction. This justifies the rationality of GFRL's design of integrating intra-field and contextual information before performing feature interaction, which is the core contribution of our work. It is worth mentioning that, MCRF-1 shows better result than GFRL, but for simplicity, we just show GFRL's training process.

4.4 Effectiveness of GFRL Variants (Q3)

We conduct ablation experiments to explore the specific contributions of FLU and Gate Unit in GFRL based on five models. FWN is proposed in NON [13], which only captures intra-field information. FLU adds contextual information

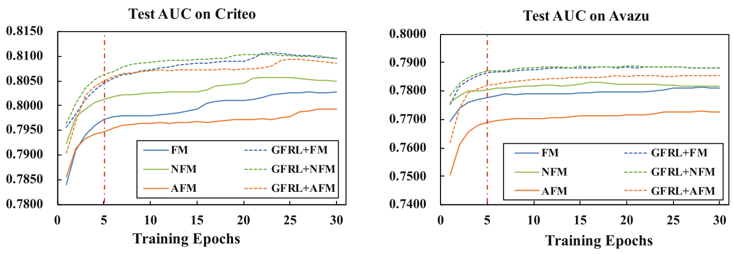


Fig. 3. The convergence curves of training process.

based on FWN. And GFRL is comprised of two FLUs and a Gate Unit. As observed from Table 3, both FLU and Gate Unit are necessary for GFRL.

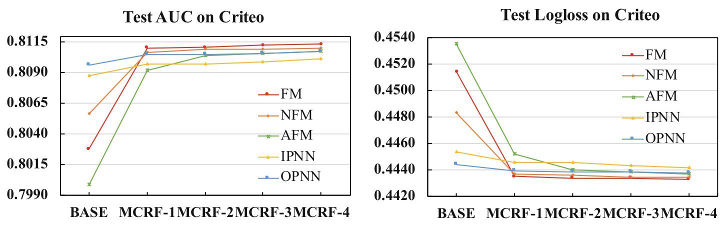
Firstly, FLU is always better than FWN, which directly certifies our idea that contextual information is necessary. The same feature has different representations in various instances by absorbing contextual information, which makes it more flexible and expressive.

Secondly, GFRL always outperforms FLU and FWN, which shows the importance of Gate Unit. It further confirms the effectiveness of the Gate Unit, which can select useful features from original and supplementary representation with learned weight in bit-level. As a comparison, both SENET and FEN learn the importance in vector-level.

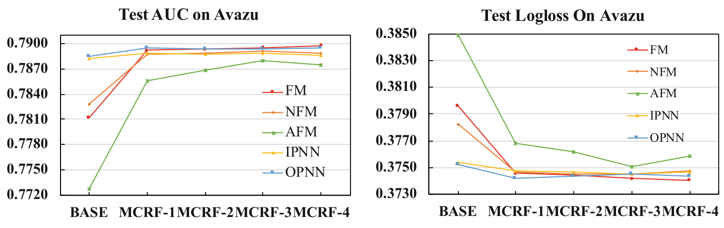
4.5 Superiority of GFRL Compared to Other Structures (Q4)

We select several modules from other models as comparisons to verify the superiority of GFRL. SENET [9] and FEN [24] are proposed to learn the importance of features. With FWN, we have four different modules: 1) SENET; 2) FWN; 3) FEN; 4) GFRL. Those are applied after the embedding layer to generate new feature representation. As shown in Table 3, we reach the following conclusions:

(1) In most cases, all four modules can boost base models' performance. It proves that refining feature representation before performing feature interaction is necessary and deserves further study. Meanwhile, from these unoptimized examples, we can know that randomly adding a module to a specific model sometimes doesn't lead to incremental results.
(2) With the help of GFRL, all the base models are improved significantly. Meanwhile, GFRL consistently outperforms other modules (FWN, SENET, FEN). Furthermore, GFRL is the only structure that improves the performance of all models. All those facts demonstrate the compatibility and robustness of GFRL, which can be well generalized in the majority of mainstream CTR prediction models to boost their performance.
(3) By checking the learning process of AFM, we find it updates slowly. This situation does not get modified either after using SENET, FWN, and FEN, or even worse. However, GFRL solves the problem, we show the training process of AFM, GFRL+AFM in Fig. 3. The possible reason is that Gate



(a) Criteo Dataset



(b) Avazu Dataset

Fig. 4. The performance of different number of channels in MCRF based upon various base models on Criteo and Avazu datasets.

Unit allows the gradient to propagate through three paths in GFRL. This illustrates the soundness and rationality of the GFRL's design again.

4.6 The Impact of Channel Numbers of MCRF (Q5)

The number of channels is the critical hyper-parameter in MCRF. In previous sections, we have shown that MCRF-1 improves the performance of several mainstream models and outperforms GFRL with one additional channel. Here, we change the number of channels from 1 to 4 to show how it affects their performance. Note that we treat the base model as #channel = 0. Fig. 4 depicted the experimental results and have the following observations:

(1) As the number of channels increases, the performance of base models is consistently improved, especially for FM, AFM, and NFM. Unlike several parallel-structured models (e.g., DeepFM, xDeepFM), MCRF performs the same operation in different channels based on discriminative feature representations generated by GFRLs. Our results verify our design that it is effective to absorb more informative feature interactions from multiple groups of embeddings.
(2) Surprisingly, FM+MCRF achieves the best performance on two datasets. Especially on the Criteo dataset, with the help of MCRF, FM and NFM always outperform IPNN and OPNN. It proves that the FM successfully captures more useful feature interactions from multiple refined representations. Furthermore, considering the results of FM+GFRL, those experiments further verify that the idea of refining feature representation is highly efficient

and reasonable. With informative representation, even simple interaction operations (such as the sum of inner product in FM) can get better results than some complicated feature interaction operations. FM+MCRF may be a good candidate for achieving state-of-the-art CTR prediction models with ease of implementation and high efficiency.

5 Conclusion

Unlike modeling feature interaction, we focus on learning flexible and informative feature representation according to the specific instance. We design a model-agnostic module named GFRL to refine raw embedding by simultaneously absorbing intra-field and contextual information. Based on GFRL, we further propose a generalized framework MCRF, which utilizes GFRLs to generate multiple groups of embedding for richer interactions without exploding the parameter space. A detailed ablation study shows that each component of GFRL contributes significantly to the overall performance. Extensive experiments show that while applying GFRL and MCRF in other CTR prediction models, better performance is always achieved, which shows the efficiency and compatibility of GFRL and MCRF. Most importantly, with the help of GFRL and MCRF, FM achieves the best performance compared to other methods for modeling complex feature interaction. Those facts identify a fundamental direction for CTR prediction tasks that it is essential to learn more informative and expressive feature representation on top of the embedding layer.

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

References

1. Cheng, H.T., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 7–10 (2016)
2. Cheng, W., Shen, Y., Huang, L.: Adaptive factorization network: learning adaptive-order feature interactions. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 3609–3616 (2020)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017)
5. He, X., Chua, T.S.: Neural factorization machines for sparse predictive analytics (2017)
6. He, X., Du, X., Wang, X., Tian, F., Tang, J., Chua, T.S.: Outer product-based neural collaborative filtering. arXiv preprint arXiv:1808.03912 (2018)
7. Hochreiter, S., urgen Schmidhuber, J., Elvezia, C.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997)

8. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7132–7141 (2018)
9. Huang, T., Zhang, Z., Zhang, J.: FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. In: Proceedings of the 13th ACM Conference on Recommender Systems, pp. 169–177 (2019)
10. Juan, Y., Zhuang, Y., Chin, W.S., Lin, C.J.: Field-aware factorization machines for CTR prediction. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 43–50 (2016)
11. Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., Sun, G.: xDeepFM: combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1754–1763 (2018)
12. Liu, B., Tang, R., Chen, Y., Yu, J., Guo, H., Zhang, Y.: Feature generation by convolutional neural network for click-through rate prediction. In: The World Wide Web Conference, pp. 1119–1129 (2019)
13. Luo, Y., Zhou, H., Tu, W., Chen, Y., Dai, W., Yang, Q.: Network on network for tabular data classification in real-world applications. arXiv preprint arXiv:2005.10114 (2020)
14. Pan, J., et al.: Field-weighted factorization machines for click-through rate prediction in display advertising. In: Proceedings of the 2018 World Wide Web Conference, pp. 1349–1357 (2018)
15. Qu, Y., et al.: Product-based neural networks for user response prediction over multi-field categorical data. ACM Trans. Inf. Syst. (TOIS) **37**(1), 1–35 (2018)
16. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000. IEEE (2010)
17. Richardson, M., Dominowska, E., Ragno, R.: Predicting clicks: estimating the click-through rate for new ads. In: Proceedings of the 16th International Conference on World Wide Web, pp. 521–530 (2007)
18. Song, W., et al.: AutoInt: automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 1161–1170 (2019)
19. Tang, W., Lu, T., Li, D., Gu, H., Gu, N.: Hierarchical attentional factorization machines for expert recommendation in community question answering. IEEE Access **8**, 35331–35343 (2020)
20. Wang, R., Fu, B., Fu, G., Wang, M.: Deep & cross network for ad click predictions. In: Proceedings of the ADKDD 2017, pp. 1–7 (2017)
21. Wu, S., et al.: TFNet: multi-semantic feature interaction for CTR prediction. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1885–1888 (2020)
22. Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., Chua, T.S.: Attentional factorization machines: learning the weight of feature interactions via attention networks. arXiv preprint arXiv:1708.04617 (2017)
23. Yang, B., Chen, J., Kang, Z., Li, D.: Memory-aware gated factorization machine for top-N recommendation. Knowl.-Based Syst. **201**, 106048 (2020)
24. Yu, Y., Wang, Z., Yuan, B.: An input-aware factorization machine for sparse prediction. In: IJCAI, pp. 1466–1472 (2019)
25. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data. In: Ferro, N., et al. (eds.) ECIR 2016. LNCS, vol. 9626, pp. 45–57. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_4