Revisiting Feature Interactions from the Perspective of Quadratic Neural Networks for Click-through Rate Prediction

HONGHAO LI, Anhui University, Hefei, Anhui, China

YIWEN ZHANG, Anhui University, Hefei, Anhui, China

YI ZHANG, Anhui University, Hefei, Anhui, China

LEI SANG, Anhui University, Hefei, Anhui, China

JIEMING ZHU, Huawei Technologies Noah's Ark Lab, Hong Kong, Hong Kong

PDF Download
3711896.3737106.pdf
17 January 2026
Total Citations: 0
Total Downloads: 1120

Published: 03 August 2025

Citation in BibTeX format

KDD '25: The 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining
August 3 - 7, 2025
Toronto ON, Canada

Conference Sponsors:
SIGMOD
SIGKDD

Revisiting Feature Interactions from the Perspective of Quadratic Neural Networks for Click-through Rate Prediction

Honghao Li
salmon1802li@gmail.com
Anhui University
Hefei, Anhui Province, China

Yiwen Zhang*
zhangyiwen@ahu.edu.cn
Anhui University
Hefei, Anhui Province, China

Yi Zhang
zhangyi.ahu@gmail.com
Anhui University
Hefei, Anhui Province, China

Lei Sang
sanglei@ahu.edu.cn
Anhui University
Hefei, Anhui Province, China

Jieming Zhu
jiemingzhu@ieee.org
Huawei Noah's Ark Lab
Shenzhen, Guangdong Province
China

Abstract

Hadamard Product (HP) has long been a cornerstone in click-through rate (CTR) prediction tasks due to its simplicity, effectiveness, and ability to capture feature interactions without additional parameters. However, the underlying reasons for its effectiveness remain unclear. In this paper, we revisit HP from the perspective of Quadratic Neural Networks (QNN), which leverage quadratic interaction terms to model complex feature relationships. We further reveal QNN's ability to expand the feature space and provide smooth nonlinear approximations without relying on activation functions. Meanwhile, we find that traditional post-activation does not further improve the performance of the QNN. Instead, mid-activation is a more suitable alternative. Through theoretical analysis and empirical evaluation of 25 QNN neuron formats, we identify a good-performing variant and make further enhancements on it. Specifically, we propose the Multi-Head Khatri-Rao Product as a superior alternative to HP and a Self-Ensemble Loss with dynamic ensemble capability within the same network to enhance computational efficiency and performance. Ultimately, we propose a novel neuron format, QNN-α, which is tailored for CTR prediction tasks. Experimental results show that QNN-α achieves new state-of-the-art performance on six public datasets while maintaining low inference latency, good scalability, and excellent compatibility. The code, running logs, and detailed hyperparameter configurations are available at: https://github.com/salmon1802/QNN.

CCS Concepts

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

*Yiwen Zhang (zhangyiwen@ahu.edu.cn) is the Corresponding author.

Keywords

Quadratic Neural Networks, Recommender Systems, CTR Prediction

ACM Reference Format:
Honghao Li, Yiwen Zhang, Yi Zhang, Lei Sang, and Jieming Zhu. 2025. Revisiting Feature Interactions from the Perspective of Quadratic Neural Networks for Click-through Rate Prediction. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3711896.3737106

1 Introduction

Hadamard Product (HP), known for its simplicity, effectiveness, and without additional parameter requirements, has long been one of the most popular explicit feature interaction methods for click-through rate (CTR) prediction tasks [7, 32, 62, 67]. By explicitly capturing bounded interactions between features, HP demonstrates strong computational efficiency and model performance in large-scale sparse data scenarios [67]. Recent studies [32, 62, 80] have further achieved significant performance improvements by leveraging HP. However, despite the encouraging results, the underlying reasons for HP's effectiveness remain unclear, motivating us to further explore the fundamental causes of HP's success.

To investigate its nature, we attempt to analyze HP from the perspective of Quadratic Neural Networks (QNN) [3, 8, 13, 74]. The element-wise multiplication operation of HP is naturally related to the core idea of QNN, which involves linearly independent quadratic polynomials χ (As shown in Figure 1). Therefore, we believe that revisiting the working mechanism of HP through the lens of QNN can provide new insights into CTR prediction. Specifically, we first derive and compare recursive formulations for MLP and HP-based CTR models from the theoretical foundation of QNN, aiming to deeply analyze the source of HP's superiority in capturing feature interactions. Through empirical evaluation and visualization analyses in Section 2.1, we observe that the nonlinear approximation ability of MLP is relatively sharp and *heavily dependent on activation functions.* In contrast, the fundamental reason for HP's effectiveness lies in its expansion of the feature space and its inherent smooth nonlinear approximation capabilities, *without explicitly relying on activation functions.* This discovery raises an intriguing

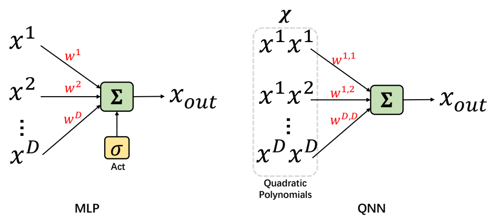


Figure 1: Comparison of MLP and QNN.

question worthy of further exploration: *Are Activation Functions Necessary for HP-based CTR Prediction?*

To answer this question, we empirically analyze 25 different neuron formats of QNN, as discussed in Section 2.2 and 2.3, claim the connections between existing CTR models and QNN, and identify the T19 variant as a strong performer. Moreover, we find that, in most cases, introducing post-activation (Post Act) into QNN reduces performance, while introducing mid-activation (Mid Act) helps QNN achieve both smooth and sharp nonlinear approximation capabilities, thereby improving performance. Based on these findings, a follow-up question naturally arises: *Is there a better QNN design for CTR prediction?*

In this paper, we deliver a positive answer to this question. Initially, we attempt to improve performance by increasing the parameter size of QNN neurons or modifying them into a bottleneck structure [19], but the results do not meet our expectations. Subsequently, we focus on the Product method for feature interactions and replace the traditional HP with a more expressive Khatri–Rao Product (KRP) [38], achieving certain performance improvements. However, since KRP requires higher computational resources compared to HP, we propose the **Multi-Head Khatri–Rao Product (MH-KRP)**. Theoretically, MH-KRP reduces the parameter size and computational complexity of KRP by a factor of H.

Furthermore, while ensemble learning is an effective method to mitigate the risk of overfitting on sparse data in CTR prediction tasks, it often incurs high computational costs [33, 41, 42, 46]. To address this issue, we propose a **Self-Ensemble Loss (SE Loss)**, which dynamically ensembles the prediction results generated by the same network during different forward passes, thereby achieving performance improvements without additional inference costs. Ultimately, we propose a novel QNN format **QNN-α** that is better suited for CTR prediction tasks. This model not only achieves state-of-the-art performance on six public datasets but also exhibits lower inference latency compared to some models that have already been successfully deployed in production environments. The core contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work to introduce QNN into CTR prediction. From the perspective of QNN, we provide new insights into feature interaction-based CTR prediction.
- We reveal the core reasons why HP improves the performance of CTR models through theoretical analysis and visualization. Moreover, we find that Post Act does not effectively enhance the performance of QNN, while Mid Act proves to be a better choice.
- We evaluate 25 QNN neuron formats, propose MH-KRP as a superior alternative to HP, and introduce SE Loss to boost performance without extra sub-networks. Ultimately, we propose QNN-α, a novel tailored format for CTR prediction.
- We conduct extensive experiments on six benchmark datasets, demonstrating the low latency, effectiveness, scalability, and compatibility of QNN-α.

1.1 Preliminaries

1.1.1 ***DEFINITION 1: CTR Prediction.*** It is a binary classification task that predicts the probability of a user clicking on an item using user profiles x_U, item attributes x_I, and context x_C features [30, 81]. A CTR input sample can be defined as a tuple: $X = \{x_U, x_I, x_C\}$, where $y \in \{0,1\}$ represents the true label of user click behavior. The goal of a CTR prediction model is to predict y and rank items based on the predicted probabilities $\hat{y}$. Most CTR models [30, 56, 57] use an embedding layer to convert X into low-dimensional dense vectors: $\mathbf{e}_i = E_i x_i$, where $E_i \in \mathbb{R}^{d\times s_i}$ is the embedding matrix, s_i is the vocabulary size for the i-th field, and d is the embedding dimension. After that, we concatenate the individual feature embeddings to get the input $\mathbf{X}_1 = \left[\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_f\right] \in \mathbb{R}^D$, where $D = \sum_{i=1}^{f} d$, and f denotes the number of feature fields. In this work, we take the input $\mathbf{X}_1$ as the first-order feature.

1.1.2 ***DEFINITION 2: Feature Interaction.*** *Implicit feature interaction* leverages MLP to automatically learn high-order feature interactions [35, 67], while *explicit feature interaction* directly models relationships between features via explicit formulas [35, 67]. A common explicit method is $\mathbf{X}_n = \mathbf{X}_1 \odot \mathbf{X}_{n-1}$ [32, 67], which uses the Hadamard Product to generate the n-th order feature $\mathbf{X}_n$.

1.1.3 ***DEFINITION 3: Quadratic Neural Networks.*** The core idea of QNN [13] is that the output of neurons in each layer depends not only on the linear transformation of input features, but also explicitly introduces quadratic interaction terms between input features [8, 72, 74]. Specifically, QNN constructs the feature space using a linearly independent set of quadratic polynomials $\chi = \{x^1x^1, x^1x^2, \dots\}$, where x^i indicate i-th variable, χ includes all possible quadratic interaction terms. This approach enhances feature relationship explicit modeling and shortens the learning path for complex interactions [32].

2 Investigation of Feature Interactions in CTR

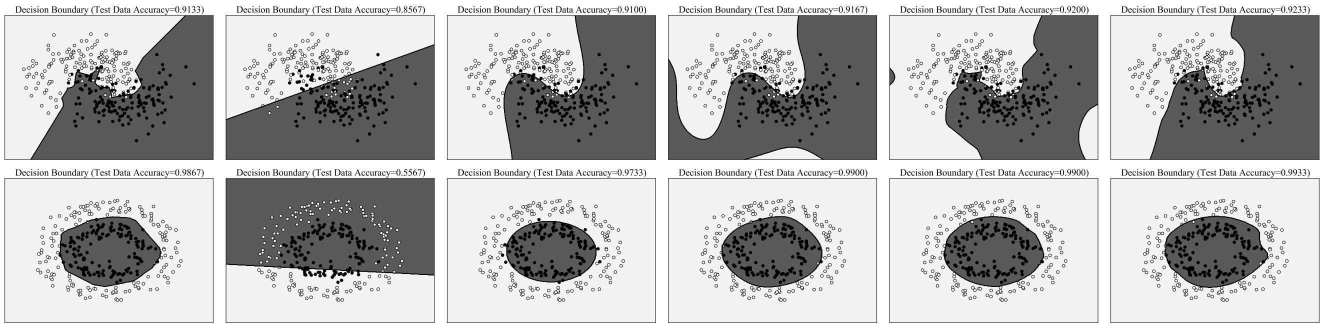
2.1 Revisiting Feature Interactions in CTR from Quadratic Neural Networks

2.1.1 ***Theoretical analysis.*** CTR prediction models typically enhance performance by integrating explicit and implicit feature interactions [35, 81]. In this paper, we revisit several classic methods for feature interactions [9, 67], which assume that explicit and implicit feature interactions are distinct yet complementary [35, 67]. Firstly, we investigate the classic DCNv2 [67] model, which employs CrossNetv2 (CNv2) for explicit feature interactions. Formally, the recursive formula in CNv2 is defined as follows[1]:

$$\mathbf{X}_{l+1} = \mathbf{X}_1 \odot \mathbf{W}_l \mathbf{X}_l, \quad l = 1, 2, \dots, L, \tag{1}$$

where $\mathbf{W}_l \in \mathbb{R}^{D\times D}$ is the learnable parameter matrix, $\mathbf{X}_l \in \mathbb{R}^D$ represents the l-th order feature, L is the total number of layers, and $\odot$ denotes the Hadamard Product. In fact, we can rewrite the

[1]The bias and residual terms are omitted for clarity.



(a) MLP **(b) MLP w/o Act** **(c) CrossNetv2 (CNv2)** **(d) T9 (ECN-like)** **(e) DCNv2=MLP+CNv2** **(f) T19**

Figure 2: Comparison of decision boundaries for 2D moon (line 1) and circle (line 2) datasets in different neuron formats.

Equation (1) as:

$$x_{l+1}^{i} = x_1^{i}\sum_{j=1}^{D} w_l^{i,j} x_l^{j} = \underbrace{w_l^{i,1}x_1^{i}x_l^{1} + w_l^{i,2}x_1^{i}x_l^{2} + \cdots + w_l^{i,D}x_1^{i}x_l^{D}}_{D \text{ interaction items}}, \quad (2)$$

where $w_l^{i,j} \in \mathbf{W}_l$ represents the learnable parameter at position (i, j), and $x_l^{j} \in \mathbf{X}_l$ denotes the j-th element of $\mathbf{X}_l$. From Equation (2), we observe that the recursive function proposed by CNv2 expands the 1-dimensional feature space into a D-dimensional feature space, where second-order polynomials $\chi = \{x_1^i x_l^1, x_1^i x_l^2, \cdots, x_1^i x_l^D\}$ introduces a nonlinear dependency between the variables x_1^i and x_l^j. Therefore, CNv2 can be regarded as a QNN. Similarly, the MLP for implicit feature interactions can be rewritten as:

$$x_{l+1}^{i} = \sigma\left(\sum_{j=1}^{D} w_l^{i,j} x_l^{j}\right) = \underbrace{\sigma\left(w_l^{i,1}x_l^{1} + w_l^{i,2}x_l^{2} + \cdots + w_l^{i,D}x_l^{D}\right)}_{D \text{ interaction items}}, \quad (3)$$

where σ represents the activation function (Act). It is evident that, without considering the activation function, although Equation (3) also contains D interaction terms, it does not include the product of multiple variables, resulting in a linear correlation with x_l^j. Therefore, the non-linear approximation ability of the MLP heavily depends on the Act [23].

ECN [32] adopts a similar method to CNv2 for modeling feature interactions. The difference lies in that the former is designed to capture exponentially growing feature interactions, while the latter can only capture linearly growing feature interactions. The recursive function of ECN-like layers, which we refer to as the T9 neuron format in this paper, can be simplified as follows:

$$\begin{aligned}
\mathbf{X}_{l+1} &= \mathbf{X}_l \odot \mathbf{W}_l\mathbf{X}_l \overset{l \geq 1}{=} \mathbf{X}_{l-1} \odot \mathbf{W}_{l-1}\mathbf{X}_{l-1} \odot \mathbf{W}_l\mathbf{X}_l,\\
x_{l+1}^{i} &\overset{l \geq 1}{=} x_{l-1}^{i}\sum_{j=1}^{D} w_{l-1}^{i,j}x_{l-1}^{j}\sum_{k=1}^{D} w_l^{i,k}x_l^{k},\\
&= \underbrace{A_{(1,1)}x_{l-1}^{1}x_l^{1} + A_{(1,2)}x_{l-1}^{1}x_l^{2} + \cdots + A_{(D,D)}x_{l-1}^{D}x_l^{D}}_{D^2 \text{ interaction items}},
\end{aligned} \quad (4)$$

where A is a simplification coefficient, defined as:

$$A_{(j,k)} = w_{l-1}^{i,j} w_l^{i,k} x_{l-1}^{i}, \quad (5)$$

From the perspective of QNN, ECN's second-order polynomials are defined as $\chi = \{x_{l-1}^1 x_l^1, x_{l-1}^1 x_l^2, \cdots, x_{l-1}^D x_l^D\}$. We observe that ECN expands the feature space from D dimensions in CNv2 to D^2 dimensions. Moreover, each interaction term exhibits a nonlinear relationship with x_{l-1}^i. As a result, ECN may have better nonlinear approximation capabilities than CNv2.

2.1.2 ***Visual Analysis of Nonlinear Approximation Ability.*** To more intuitively analyze and compare the approximation capabilities between QNN and MLP, we visualize the decision boundaries of various neuron formats on the widely-used 2D moon and circle test dataset[2] [48] as shown in Figure 2. We observe that, except for Figure 2 (b), all other neuron formats exhibit nonlinear approximation capabilities, which aligns with our expected results and validates the correctness of our theoretical analysis.

Furthermore, MLP shows sharper decision boundaries compared to QNN variants, CNv2 and T9. This sharpness may be due to the fact that the ReLU function is not smooth at zero [43, 52]. Meanwhile, CNv2 and T9 exhibit extremely smooth nonlinear decision boundaries, with T9 demonstrating accuracy surpassing that of MLP on two datasets. Thus, we conclude that relying solely on the nonlinear approximation capability of ReLU often results in sharper decision boundaries. In contrast, QNN inherently possesses a smooth nonlinear approximation capability, which may potentially help reduce misclassifications near the boundaries.

Besides, we observe that DCNv2, which combines both MLP and CNv2, blends the sharp and smooth decision boundaries, achieving higher accuracy. This performance enhancement has been validated by previous works [35, 67], where explicit (e.g., CNv2) and implicit (e.g., MLP) feature interactions are combined to leverage the complementary strengths of different interaction methods. In Figure 2, this abstract concept of explicit and implicit can be further intuitively understood: there is often a need for sharp boundaries with fine localized carving, as well as boundaries that allow for globally smooth transitions.

2.2 Necessity of Activation Functions

The above analysis demonstrates that QNN possesses nonlinear approximation capabilities even without the use of activations. The

[2]For hyperparameter configurations, we uniformly set the hidden dimension size to 20, the learning rate to 1e-2, the number of layers to 2, and training 500 epochs on moon dataset and 100 epochs on circle dataset.

Table 1: Performance comparison of QNN on Criteo dataset. CTR researchers consider a *0.1%* improvement in Logloss and AUC as statistically significant [81].

Type	Neuron Format	Post Act: $\mathbf{ReLU}(\Phi(\mathbf{X}))$		Time×Epochs	W/O Post Act: $\Phi(\mathbf{X})$		Time×Epochs	#Params	Gap	
		Logloss↓	AUC(%)↑		Logloss↓	AUC(%)↑			△Logloss	△AUC
MLP [9]	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X}$	0.4381	81.37	2min × 11	0.4568	79.32	2min × 12	15.14M	+0.0187	-2.05
CrossNetv2 [67]	$\Phi(\mathbf{X}) = \mathbf{X}_1 \odot \mathbf{W}_a\mathbf{X} + \mathbf{X}$	0.4389	81.34	2min × 12	0.4385	81.36	2min × 10	15.74M	-0.0004	+0.02
T1 (FM-like [42, 53])	$\Phi(\mathbf{X}) = \mathbf{X}^\intercal\mathbf{W}_a\mathbf{X} + \mathbf{W}_b\mathbf{X}$	0.4395	81.27	24min × 11	0.4388	81.36	24min × 10	17.66M	-0.0007	+0.09
T2	$\Phi(\mathbf{X}) = \mathbf{X}^\intercal\mathbf{W}_a\mathbf{X}$	0.4391	81.32	23min × 10	0.4439	81.30	23min × 14	17.63M	+0.0048	-0.02
T3	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X}^2$	0.4401	81.22	2min × 16	0.4412	81.26	2min × 31	15.74M	+0.0011	+0.04
T4	$\Phi(\mathbf{X}) = (\mathbf{W}_a\mathbf{X})^2$	0.4382	81.39	2min × 16	0.4381	81.41	2min × 12	15.74M	-0.0001	+0.02
T5 (PEPNet-like [4])	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X}$	0.4383	81.39	2min × 11	0.4385	81.37	2min × 9	16.91M	+0.0002	-0.02
T6	$\Phi(\mathbf{X}) = \mathbf{X}^\intercal\mathbf{W}_a\mathbf{X} + \mathbf{W}_b\mathbf{X}^2$	0.4416	81.28	23min × 13	0.4468	81.22	23min × 15	17.66M	+0.0052	-0.06
T7	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} + \mathbf{W}_c\mathbf{X}^2$	0.4391	81.28	2.5min × 11	0.4403	81.24	2.5min × 11	18.08M	+0.0012	-0.04
T8	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} + \mathbf{W}_c\mathbf{X}$	0.4384	81.37	2.5min × 11	0.4384	81.37	2.5min × 13	18.08M	0.0000	0.00
T9 (ECN-like [32])	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a\mathbf{X} + \mathbf{X}$	**0.4369**	**81.51**	2.5min × 17	**0.4369**	**81.53**	2.5min × 16	15.74M	0.0000	+0.02
T10	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a\mathbf{X} + \mathbf{W}_b\mathbf{X}$	0.4381	81.39	2.5min × 10	0.4373	81.50	2.5min × 19	16.91M	-0.0008	+0.11
T11	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \mid\mid (\mathbf{W}_b\mathbf{X})^2$	0.4381	81.38	2min × 12	0.4378	81.44	2min × 12	15.74M	-0.0003	+0.06
T12	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} \mid\mid \mathbf{W}_c\mathbf{X}^2$	0.4381	81.40	2.5min × 12	0.4384	81.35	2.5min × 12	16.32M	+0.0003	-0.05
T13 (FINAL-like [80])	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} \mid\mid \mathbf{W}_c\mathbf{X}$	0.4386	81.35	2min × 11	0.4382	81.39	2min × 12	16.32M	-0.0004	+0.04
T14	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} \mid\mid \mathbf{W}_b\mathbf{X}$	0.4387	81.34	2.5min × 15	0.4379	81.42	2.5min × 12	15.74M	-0.0008	+0.08
T15	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \mid\mid (\mathbf{W}_a\mathbf{X})^2$	0.4377	81.42	2min × 15	0.4378	81.44	2min × 13	15.15M	+0.0001	+0.02
T16 (GCN-like [64])	$\Phi(\mathbf{X}) = \mathbf{X} \odot (\mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X}) + \mathbf{X}$	0.4379	81.43	2.5min × 17	0.4378	81.48	2.5min × 14	16.91M	-0.0001	+0.05
T17	$\Phi(\mathbf{X}) = \mathbf{X} \odot (\mathbf{W}_a\mathbf{X} + \mathbf{W}_b\mathbf{X}) + \mathbf{X}$	0.4384	81.35	2.5min × 13	0.4372	81.50	2.5min × 15	16.91M	-0.0012	+0.15
T18	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a\mathbf{X} \mid\mid \mathbf{X}$	0.4392	81.27	2.5min × 10	0.4388	81.33	2.5min × 10	18.08M	-0.0004	+0.06
T19 (as Figure 2 (f))	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{ReLU}(\mathbf{W}_a\mathbf{X}) + \mathbf{X}$	0.4377	81.46	2.5min × 18	**0.4365**	**81.56**	2.5min × 26	15.74M	-0.0012	+0.10
T20	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a\mathbf{X} + \mathbf{W}_a\mathbf{X} + \mathbf{X}$	0.4385	81.34	2.5min × 14	0.4394	81.26	2.5min × 13	15.74M	+0.0009	-0.08
T21	$\Phi(\mathbf{X}) = (\mathbf{X} \odot \mathbf{W}_a\mathbf{X})^2 + \mathbf{X}$	0.4430	81.37	2.5min × 30	0.4382	81.40	2.5min × 22	15.74M	-0.0048	+0.03
T22	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a\mathbf{X} + \alpha \odot \mathbf{X}$	**0.4369**	**81.51**	2.5min × 19	**0.4369**	**81.53**	2.5min × 16	15.74M	0.0000	+0.02
T23	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{W}_b\mathbf{X} + \mathbf{X}$	0.4379	81.42	2.5min × 10	0.4384	81.41	2.5min × 10	16.91M	+0.0005	-0.01
T24 (MaskNet-like [68])	$\Phi(\mathbf{X}) = \mathbf{W}_a\mathbf{X} \odot \mathbf{ReLU}(\mathbf{W}_b\mathbf{X}) + \mathbf{X}$	0.4378	81.42	2.5min × 12	0.4381	81.42	2.5min × 12	16.91M	+0.0003	0.00
T25	$\Phi(\mathbf{X}) = \mathbf{X} \odot \mathbf{W}_a(\mathbf{ReLU}(\mathbf{W}_b\mathbf{X})) + \mathbf{X}$	0.4378	81.43	2.5min × 14	0.4378	81.47	2.5min × 14	15.74M	0.0000	+0.04

question then arises: *does the introduction of Activations further enhance the network's approximation ability?* To answer this question, we conduct experiments[3] on the well-known large-scale Criteo [81] dataset. The experimental results are shown in Table 1, where we refer to the classic QNN formats from [74] and further extend them to 25 types. Meanwhile, we find that some formats correspond to existing CTR model designs, such as T1, T5, T9, T13, and T16.

Surprisingly, the $\Phi(\mathbf{X})$ variant outperforms $\mathbf{ReLU}(\Phi(\mathbf{X}))$ in 64% of cases (in blue) in terms of AUC, shows comparable performance in 8% of cases (in green), and the former performs worse than the latter in 28% of cases (in red). Even in the few cases where performance degradation occurs (e.g., T2, T6, and T7), the impact is minimal and significantly smaller than the performance loss observed when Post Act is removed in MLP. This indicates that QNN's inherent approximation ability is sufficient, making the use of Post Act unnecessary since it brings none or negligible benefits. In fact, Andrew *et al.* [40] have already pointed out that Act may lead to a certain degree of information loss due to its zero output for negative inputs, further highlighting the potential of QNN in CTR prediction tasks. Additionally, we observe that the training costs of networks with and without the Post Act have their advantages and disadvantages. For instance, in T4, the latter outperforms the former, while in T3, the opposite is true. Therefore, we conclude that the **Post Act** is unnecessary for QNN.

2.3 Finding An Effective QNN Design

Given the wide variety of neuron formats in QNN, a natural question arises: *which QNN format is effective for CTR prediction tasks?* To answer this question, we observe from Table 1 that, without using Post Act, QNN consistently outperforms MLP, with several neuron formats achieving AUC levels of up to 81.50 (significantly surpassing MLP). Among these high-performing neuron formats, T19 achieves the best performance. Its structure is similar to T9, but the key difference is that T19 introduces **Mid Act**, which we define as activations applied within a single QNN layer before Hadmard Product. From the longer training epochs required for T19, we observe that T19 continues to improve as training progresses, whereas other QNN variants (e.g., CrossNetv2, T9, and T22) tend to overfit at earlier stages. Thus, we identify T19 as an effective neuron format and show that Mid Act enhances QNN's generalization. Moreover, T19 in Figure 2 also exhibits a combination of sharp and smooth decision boundaries similar to DCNv2, achieving the highest accuracy. This indicates that introducing **Mid Act** into QNN enables it to integrate the sharp approximation capability brought by Act with its inherent smooth approximation capability, thereby achieving a complementary overall approximation.

Besides, several pairs of neuron formats are worth noting. For example, [T9 vs T23 and T19 vs T24] indicate that adding more parameters may not improve model performance, which is contrary to observations in computer vision [19, 39]. [T9 vs T18] show that using the sum operation to introduce residual connections is superior to the concat operation. [T1 vs T2 and T5 vs T23] demonstrate the importance of residual connections for QNN. [MLP, T1, T3, and T4] reveal that even the simple introduction of quadratic terms in traditional linear layers can reduce MLP's reliance on Act. [T1, T5, T9, T13, T16, and T24] are all representative baseline models designed by CTR researchers and can be regarded as specific types of QNN. Therefore, understanding feature interaction modeling in CTR from the perspective of QNN may provide new insights for

[3]All networks share the same hyperparameters, with 3 layers fixed. △AUC = AUC in $\Phi(\mathbf{X})$ - AUC in $\mathbf{ReLU}(\Phi(\mathbf{X}))$, and △Logloss is defined similarly. The **Gap** column compares performance with and without Post Act. To prevent overfitting, we employ early stopping with a patience value of 2. Further details are in our open-source code and logs.

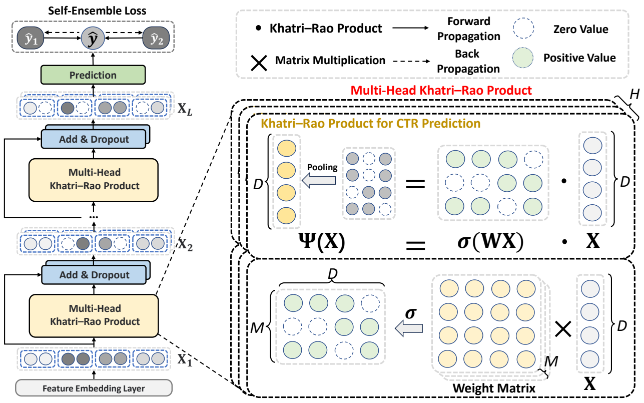


Figure 3: Architecture of the QNN-α.

future research. We recommend readers to refer to [3, 8, 10, 12, 13] for a broader and deeper understanding of QNN.

3 A Better QNN Design for CTR prediction

Based on the conclusions in Section 2, we believe that QNN holds significant potential for improving CTR model performance but remains underexplored. Therefore, in this section, we aim to further understand the underlying mechanisms of QNN neuron formats and make improvements to fully leverage the powerful approximation capabilities of QNN, thus proposing QNN-α as shown in Figure 3.

3.1 Khatri–Rao Product: A Better Alternative to the Hadamard Product

Hadamard Product (HP) has long been one of the most well-known methods for modeling feature interactions [22, 32, 54, 67], demonstrating strong performance across various domains [39, 72, 74] (as evidenced by the T19's superior performance). From the perspective of QNN, the effectiveness of HP lies in its ability to element-wisely expand the feature space and introduce nonlinear second-order polynomials χ. For example, in Equation (1), HP expands the feature space from 1 dimension to D dimensions, while in Equation (4), it extends it to D^2 dimensions. However, we argue that HP's lack of scalability limits the model capacity. To better understand this, we first revisit a simple HP:

$$\mathbf{X}_a \odot \mathbf{X}_b = x_a^i x_b^i = \{x_a^1 x_b^1, x_a^2 x_b^2, \dots, x_a^D x_b^D\}. \tag{6}$$

It is evident that HP requires two matrices of the same shape, which forces $\mathbf{W}_l$ in Equation 4 to always maintain a $D \times D$ scale, thereby reducing the network's scalability[4].

To address this issue, we initially attempt to introduce additional parameters or bottleneck structures to the neuron, as shown in [T9 vs T17 and T19 vs T25] in Table 1. Unfortunately, this attempt reduces performance. we realize that this may be due to HP's overly simplistic method to handling information interactions in high-dimensional feature spaces, which fails to adapt to the diversity and complexity of feature distributions, thereby limiting the model's capacity. Subsequently, we find that the Khatri–Rao Product (KRP) [38] is a more expressive method for modeling feature interactions. Specifically, KRP performs a Kronecker Product [2] on each column of two feature matrices, generating a higher-dimensional interaction matrix. For easy understanding, here is a simple example:

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{A}^1 & \mathbf{A}^2 \end{array}\right] = \left[\begin{array}{c|c} a^{1,1} & a^{1,2} \end{array}\right],$$
$$\mathbf{B} = \left[\begin{array}{c|c} \mathbf{B}^1 & \mathbf{B}^2 \end{array}\right] = \left[\begin{array}{c|c} b^{1,1} & b^{1,2} \\ b^{2,1} & b^{2,2} \end{array}\right],$$
$$\mathbf{A} \bullet \mathbf{B} = \left[\begin{array}{c|c} \mathbf{A}^1 \otimes \mathbf{B}^1 & \mathbf{A}^2 \otimes \mathbf{B}^2 \end{array}\right] = \left[\begin{array}{c|c} a^{1,1}b^{1,1} & a^{1,2}b^{1,2} \\ a^{1,1}b^{2,1} & a^{1,2}b^{2,2} \end{array}\right] = \mathbf{C}, \tag{7}$$

where $\bullet$ denotes KRP, and $\otimes$ represents the Kronecker Product. From Equation (7), we observe that when $\mathbf{B} = b^{(1,:)}$, KRP reduces to the HP. Therefore, we can consider KRP as a generalized version of HP. Furthermore, when $\mathbf{A} \in \mathbb{R}^D$ and $\mathbf{B} \in \mathbb{R}^{M \times D}$, the resulting $\mathbf{C} \in \mathbb{R}^{M \times D}$. However, in CTR prediction, following the CrossNetv2 paradigm, we aim for the neuron's output to have the same dimensions as $\mathbf{X}$ to facilitate deeper computations and the introduction of residual connections. Therefore, dimensionality reduction of $\mathbf{C}$ is necessary, and sum pooling is a suitable choice for this purpose. Finally, we redefine KRP for CTR prediction as:

$$\begin{aligned} \mathbf{A} \bullet \mathbf{B} &= \left[\begin{array}{c|c} \mathbf{A}^1 \otimes \mathbf{B}^1 & \mathbf{A}^2 \otimes \mathbf{B}^2 \end{array}\right]_{\text{sum}} \\ &= \left[\begin{array}{c|c} a^{1,1}b^{1,1} + a^{1,1}b^{2,1} & a^{1,2}b^{1,2} + a^{1,2}b^{2,2} \end{array}\right] = \mathbf{C}. \end{aligned} \tag{8}$$

If KRP is applied to T9, its neuron format is as follows:

$$\begin{aligned} \mathbf{X}_{l+1} &= \mathbf{X}_l \bullet \mathbf{W}_l \mathbf{X}_l, \quad l = 1, 2, \dots, L, \\ x_{l+1}^i &\overset{l \geq 1}{=} x_{l-1}^i \sum_{j=1}^{D} \sum_{p=1}^{M} w_{l-1}^{p,i,j} x_{l-1}^j \sum_{k=1}^{D} \sum_{q=1}^{M} w_l^{q,i,k} x_l^k, \\ &= \underbrace{A_{(1,1)} x_{l-1}^1 x_l^1 + A_{(1,2)} x_{l-1}^1 x_l^2 + \dots + A_{(D,D)} x_{l-1}^D x_l^D}_{D^2 \text{ interaction items}}, \end{aligned} \tag{9}$$

where $\mathbf{W}_l \in \mathbb{R}^{M \times D \times D}$, and $A_{(j,k)} = \sum_{p=1}^{M} \sum_{q=1}^{M} w_{l-1}^{p,i,j} w_l^{q,i,k} x_{l-1}^i$. In this way, we can set M as a hyperparameter to adjust the model capacity and enhance scalability.

3.2 Multi-Head Khatri–Rao Product

Theoretically, while KRP outperforms HP, it requires more computational resources, which is another issue we need to address. Considering the success of Transformers [63], multi-head attention mechanisms have been widely used to reduce computational costs while achieving performance improvements. We incorporate this idea into KRP-based QNN and propose a new QNN-α version, whose neuron format is defined as follows:

$$\begin{aligned} \Psi(\mathbb{X}^h) &= \mathbb{X}^h \bullet \mathbf{ReLU}(\mathbb{W}_a^h \mathbf{X}) + \mathbb{X}^h, \\ \Phi(\mathbf{X}) &= \Psi(\mathbb{X}^1) \mathbin{||} \Psi(\mathbb{X}^2) \mathbin{||} \dots \mathbin{||} \Psi(\mathbb{X}^H), \end{aligned} \tag{10}$$

where $\mathbb{X}^h \in \mathbb{R}^{\mathbb{D}}$ denotes the input of the h-th subspace, $\mathbb{W}_a \in \mathbb{R}^{M \times \mathbb{D} \times D}$, $\mathbb{D} = D/H$, H denotes the number of heads, and $||$ denotes concatenation. This theoretically reduces both the parameter size and computational complexity of QNN-α by a factor of H, decreasing from $O(MD^2)$ to $O\left(MD^2/H\right)$.

[4]In this paper, we define scalability as the network's capability for horizontal scaling.

3.3 Self-Ensemble Loss

While the Khatri-Rao Product effectively enhances the model's expressiveness and capacity, highly expressive models often exhibit instability when handling sparse data, increasing the risk of overfitting [21]. To mitigate this, we draw inspiration from ensemble learning [11], a widely adopted technique in CTR prediction [16, 32, 33, 67]. This method effectively reduces the prediction bias of a single network on specific samples by ensembling the predictions of multiple sub-networks, thereby enhancing overall generalization ability [6, 16]. However, directly relying on multiple sub-networks for ensembling not only increases computational and storage costs but also imposes higher resource demands for model deployment. To address this, inspired by the ideas of R-Drop [69] and Tri-BCE [32], we propose a **Self-Ensemble Loss (SE Loss)**, which eliminates the need for additional sub-networks. Instead, *it dynamically forms an ensemble by aggregating multiple stochastic outputs from a single network during forward propagation, where the stochasticity is introduced by the use of dropout [58].* Meanwhile, it introduces a consistency constraint to reduce the discrepancy between the ensemble and individual forward results, thereby mitigating the prediction bias of the network. Let $\hat{y}_1 = \text{Forward}_1(X), \hat{y}_2 = \text{Forward}_2(X), \hat{y} = (\hat{y}_1 + \hat{y}_2)/2$, the loss is formulated in the following form:

$$\mathcal{L}_{SE} = -\frac{1}{N}\sum_{i=1}^{N}\left[\tilde{y}_i \log\left(\hat{y}_{1,i}\hat{y}_{2,i}\right) + (1-\tilde{y}_i)\log\left(\left(1-\hat{y}_{1,i}\right)\left(1-\hat{y}_{2,i}\right)\right)\right], \tag{11}$$

where y denotes the true labels, $\tilde{y}$ represents the frozen $\hat{y}$, N denotes the batch size, and the final loss is $\mathcal{L}_{total} = \mathcal{L}_{CTR} + \mathcal{L}_{SE}$, where $\mathcal{L}_{CTR} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(\hat{y}_i\right) + (1-y_i)\log\left(1-\hat{y}_i\right)\right)$. Intuitively, $\mathcal{L}_{SE}$ forces the prediction results of two forward passes, $\hat{y}_{1,i}$ and $\hat{y}_{2,i}$, to simultaneously approach 1 or 0, thereby ensuring that the predictions of the two forwards remain consistent at the sample level. In this way, $\mathcal{L}_{SE}$ jointly constrains and optimizes the different prediction results of the single network, enabling mutual correction among the predictions. This reduces potential overfitting or bias issues in the network. Moreover, unlike the hard label supervision brought by y in $\mathcal{L}_{CTR}$, $\mathcal{L}_{SE}$ introduces richer soft label information $\tilde{y}$ to the network. This allows the QNN to learn smoother decision boundaries during training, which is more conducive to handling complex distributions and challenging-to-classify samples [44]. Theoretically, we can perform more forward passes on the single network to obtain richer soft label information. However, this exponentially increases the training cost, which is unacceptable in industrial scenarios. Moreover, our experiments show that additional forward pass do not lead to significant performance improvements. Therefore, we recommend using only two forward passes. Notably, this idea can be understood or further improved from multiple perspectives, such as contrastive learning and knowledge distillation. We recommend referring to [17, 25, 31, 60, 69] to help readers gain deeper technical insights.

4 Experiments

In this section, we conduct comprehensive experiments on six CTR prediction datasets to validate the effectiveness, efficiency, scalability, and compatibility of our proposed QNN-α.

Table 2: Dataset statistics

Dataset	#Instances	#Fields	#Features
Tenrec	120,342,306	15	3,404,996
Criteo	45,840,617	39	910,747
ML-1M	739,012	7	9,751
Frappe	288,609	10	5,383
iPinYou	19,495,974	16	665,765
KKBox	7,377,418	13	91,756

4.1 Experiment Setup

4.1.1 ***Datasets.*** We evaluate QNN-α on six CTR prediction datasets: Tenrec[5] [75], Criteo[6] [81], ML-1M[7] [57], Frappe[8] [1, 7], iPinYou[9] [51], and KKBox[10] [79]. Table 2 provides detailed information about these datasets. A more detailed description of these datasets can be found in the given references and links. We follow the approach outlined in [81]. For the Criteo dataset, we discretize the numerical feature fields by rounding down each numeric value x to $\lfloor \log^2(x) \rfloor$ if $x > 2$, and $x = 1$ otherwise. We set a threshold to replace infrequent categorical features with a default "OOV" token. We set the threshold to 10 for Criteo, KKBox, and Tenrec, 2 for iPinYou, and 1 for the small dataset ML-1M and Frappe. More specific data processing procedures and results can be found in our run logs[11].

4.1.2 ***Evaluation Metrics.*** To compare the performance, we utilize two commonly used metrics in CTR models: **Logloss**, **AUC** [31, 62, 78]. AUC stands for Area Under the ROC Curve, which measures the probability that a positive instance will be ranked higher than a randomly chosen negative one. Logloss is the result of the calculation of $\mathcal{L}_{CTR}$. A lower Logloss suggests a better capacity for fitting the data. Besides, **Latency** is an important evaluation metric in CTR prediction. Therefore, we calculate the average inference latency per 100 samples on the test set.

4.1.3 ***Baselines.*** We compared QNN-α with some SOTA models. Further, we select several high-performance CTR representative baselines, such as PNN [50] and Wide & Deep [6] (2016); DeepFM [16] and DCNv1 [66] (2017); xDeepFM (2018) [35] (2018); AutoInt* (2019) [57]; AFN* (2020) [7]; DCNv2 [67] and EDCN [5], MaskNet [68] (2021); CL4CTR [65], EulerNet [61], FinalMLP [42], FINAL [80] (2023); RFM [62], ECN [32] (2024).

4.1.4 ***Implementation Details.*** We implement all models using PyTorch [47] and refer to existing works [24, 81]. We employ the Adam optimizer [28] to optimize all models, with a default learning rate set to 0.001. For the sake of fair comparison, we set the embedding dimension to 128 for KKBox and 16 for the other datasets [79, 81]. The batch size is set to 4,096 on the ML-1M, Frappe, and iPinYou datasets and 10,000 on the other datasets. To prevent overfitting, we employ early stopping with a patience value of 2. During training, we employ a Reduce-LR-on-Plateau scheduler that reduces

[5]https://static.qblv.qq.com/qblv/h5/algo-frontend/tenrec_dataset.html
[6]https://www.kaggle.com/c/criteo-display-ad-challenge
[7]https://grouplens.org/datasets/movielens
[8]http://baltrunas.info/research-menu/frappe
[9]https://contest.ipinyou.com/
[10]https://www.kkbox.com/intl
[11]https://github.com/salmon1802/QNN/tree/main/checkpoints

Table 3: Performance comparison of different deep CTR models. "*": Integrating the original model with DNN networks. Meanwhile, we conduct a two-tailed T-test to assess the statistical significance between our models and the best baseline (⋆: p < 1e-3). *Abs.Imp* represents the absolute performance improvement of QNN over the strongest baseline. Typically, CTR researchers consider an improvement of *0.001 (0.1%)* in Logloss and AUC to be statistically significant [5, 65, 66, 81].

Models	Tenrec		Criteo		ML-1M		Frappe		iPinYou		KKBox	
	Logloss↓	AUC(%)↑	Logloss↓	AUC(%)↑	Logloss↓	AUC(%)↑	Logloss↓	AUC(%)↑	Logloss↓	AUC(%)↑	Logloss↓	AUC(%)↑
DNN [9]	0.4366	80.21	0.4380	81.40	0.3100	90.30	0.1653	98.11	0.005545	78.06	0.4811	85.01
PNN [50]	0.4358	80.28	0.4378	81.42	0.3070	90.42	0.1405	98.45	0.005544	78.13	0.4793	85.15
Wide & Deep [6]	0.4366	80.22	0.4376	81.42	0.3056	90.45	0.1525	98.32	0.005542	78.09	0.4852	85.04
DeepFM [16]	0.4365	80.23	0.4375	81.43	0.3073	90.51	0.1575	98.37	0.005549	77.94	0.4785	85.31
DCNv1 [66]	0.4360	80.26	0.4376	81.44	0.3156	90.38	0.1544	98.38	0.005541	78.13	0.4766	85.31
xDeepFM [35]	0.4363	80.24	0.4376	81.43	0.3054	90.47	0.1509	98.45	0.005534	78.25	0.4772	85.35
AutoInt* [57]	0.4362	80.25	0.4390	81.32	0.3112	90.45	0.1520	98.41	0.005544	78.16	0.4773	85.34
AFN* [7]	0.4357	80.30	0.4384	81.38	0.3048	90.53	0.1598	98.19	0.005539	78.17	0.4842	84.89
DCNv2 [67]	0.4359	80.28	0.4376	81.45	0.3098	90.56	0.1484	98.45	0.005539	78.26	0.4787	85.31
EDCN [5]	0.4356	80.30	0.4386	81.36	0.3073	90.48	0.1620	98.41	0.005573	77.93	0.4952	85.27
MaskNet [68]	0.4363	80.23	0.4387	81.34	0.3080	90.34	0.1916	98.32	0.005556	77.85	0.5003	84.79
CL4CTR [65]	0.4363	80.24	0.4383	81.35	0.3074	90.33	0.1559	98.27	0.005543	78.06	0.4972	83.78
EulerNet [61]	0.4362	80.24	0.4379	81.47	0.3050	90.44	0.1331	98.50	0.005540	78.30	0.4922	84.27
FinalMLP [42]	0.4359	80.28	0.4373	81.45	0.3058	90.52	0.1472	98.51	0.005556	78.02	0.4822	85.10
FINAL(2B) [80]	0.4355	80.31	0.4371	81.49	0.3035	90.53	0.1392	98.47	0.005540	78.13	0.4800	85.14
RFM [62]	0.4358	80.29	0.4374	81.47	0.3048	90.51	0.1476	98.48	0.005540	78.25	0.4853	84.70
ECN [32]	0.4361	80.23	0.4364	81.55	0.3013	90.59	0.1587	98.49	0.005534	78.43	0.4778	85.40
QNN-α	**0.4349***	**80.35***	**0.4358***	**81.63***	**0.2960***	**90.87***	**0.1313***	**98.62***	**0.005516***	**78.63***	**0.4730***	**85.76***
Abs.Imp	-0.0006	+0.04	-0.0006	+0.08	-0.0053	+0.28	-0.0018	+0.11	-0.000018	+0.20	-0.0036	+0.36

the learning rate by a factor of 10 when performance stops improving in any given epoch [79, 81]. The hyperparameters of the baseline model are configured and fine-tuned based on the *optimal values* provided in [24, 81] and their original paper. All experiments in this paper are conducted under the same environment using NVIDIA GeForce RTX 4090 GPU. Further details on model hyperparameters and dataset configurations are available in our straightforward and accessible running logs[11].

4.2 Overall Performance

To validate the effectiveness of QNN-α, we conduct a performance comparison across six datasets. The experimental results are presented in Table 3, where bold numbers indicate the best performance, and underlined numbers represent the second-best performance. We can draw the following conclusions:

- Ensemble models consistently achieve better performance. For example, DCNv2, FinalMLP, and FINAL(2B) show significant performance improvements compared to a standalone DNN.
- QNN-α achieves a new SOTA across all datasets with only a single network.
- Models categorized as QNN, like FINAL(2B) and ECN, show strong performance, ranking second on Tenrec, Criteo, ML-1M, iPinYou, and KKBox, validating their effectiveness.
- After thorough tuning, the performance differences between models on the Tenrec and Criteo datasets become minimal, which is also reported in [81]. This highlights the difficulty of achieving performance gains on large-scale sparse datasets. Meanwhile, minor gains on open datasets may yield significant improvements in industrial applications [42].

4.3 In-Depth Study of QNN-α

4.3.1 ***Ablation Study.*** To investigate the impact of each component of QNN-α on its performance, we conduct experiments on

Table 4: Ablation study of QNN-α.

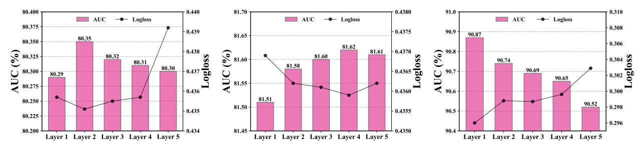
Model	Criteo		ML-1M		Frappe		KKBox	
	Logloss ↓	AUC(%) ↑	Logloss ↓	AUC(%) ↑	Logloss ↓	AUC(%) ↑	Logloss ↓	AUC(%) ↑
w/o Res	\	\	0.2971	90.78	0.1468	98.51	\	\
w/o Mid Act	0.4362	81.57	0.2969	90.79	0.1453	98.49	0.4773	85.57
w/o KRP	0.4380	81.38	0.3016	90.66	0.1630	98.51	0.4901	84.66
w/o SE Loss	0.4363	81.57	0.2970	90.78	0.1581	98.45	0.4783	85.57
QNN-α	0.4358	81.63	0.2960	90.87	0.1313	98.62	0.4730	85.76

several variants of QNN-α: (1) w/o Res: removes the residual term in Equation (10), (2) w/o Mid Act: removes the activation function before the KRP operation in Equation (10), (3) w/o KRP: replaces the KRP in Equation (10) with a linear transformation, (4) w/o SE Loss: removes the Self-Ensemble Loss.

Table 4[12], we observe that all variants of QNN experience some degree of performance degradation, which indicates the necessity of each component. w/o Res highlights the importance of the residual term for QNN. Removing it leads to gradient explosion on some datasets, making the model unable to train properly. w/o Mid Act shows that while post-activation (Post Act) reduces QNN's performance, placing the activation function in an appropriate position further improves the network's performance. w/o KRP contributes the most to performance loss, with a degradation of over 1% on the KKBox dataset, demonstrating the effectiveness of KRP. w/o SE Loss confirms that the two forward passes and the consistency constraints contribute to the model's performance improvements.

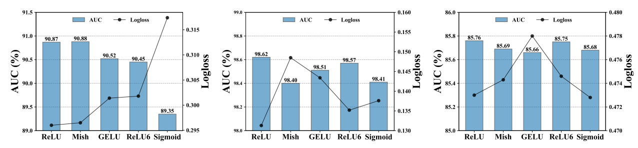
4.3.2 ***Impact of Different Network Depths in QNN-α.*** The depth of a neural network is often an important hyperparameter [19]. To investigate the impact of network depth L on the performance of QNN-α, we conduct experiments with different depths on the Tenrec, Criteo, and ML-1M datasets. As shown in Figure 4, QNN-α achieves optimal performance with 2-layer, 4-layer, and 1-layer

[12] "\" indicates that the network fails to train success.



(a) Tenrec (b) Criteo (c) ML-1M

Figure 4: Performance of different L in QNN-α.



(a) ML-1M (b) Frappe (c) KKBox

Figure 5: Performance of different Mid Act in QNN-α.

settings on the Tenrec, Criteo, and ML-1M datasets, respectively, followed by a performance decline. This indicates that networks that are either too deep or too shallow can result in performance degradation. We recommend setting the search space for L to {1, 2, 3, 4} during hyperparameter tuning as a good practice.

4.3.3 *Impact of Different Mid Act in QNN-α.*

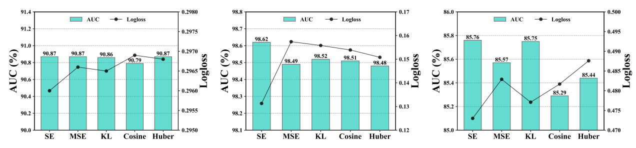
In the design of our QNN-α, we follow most CTR models [35, 67, 80], where the ReLU activation function is used as the Mid Act. To verify the reasonableness of this design, we further experiment with various activation functions. The experimental results, shown in Figure 5, indicate that ReLU demonstrates competitive performance across multiple datasets. This suggests that using a more complex Act does not provide additional performance improvements for QNN, especially those without a zero-value region (e.g., Sigmoid).

4.3.4 *Impact of Different Loss in QNN-α.*

To validate the effectiveness of the proposed SE Loss, we compare it with several commonly used consistency losses. Specifically, we replace only the loss function without altering the computation process of $\hat{y}, \hat{y}_1$, and $\hat{y}_2$. The experimental results, shown in Figure 6, demonstrate that SE Loss achieves the best performance across all three datasets. For example, on the ML-1M dataset, while AUC values for the five losses are similar, SE Loss achieves the lowest Logloss, demonstrating its effectiveness in improving classification capability [36]. On the KKBox dataset, SE Loss also outperforms KL Loss, further supporting this conclusion. On the Frappe dataset, SE Loss shows a clear advantage, with both AUC and Logloss significantly better than other loss functions, including a notable Logloss reduction at the 0.01 level. These results underscore the strong generalization and optimization performance of SE Loss. Additionally, we recommend setting the dropout rate to 0.1, as it works well across all datasets in our experiments, while higher rates consistently degrade performance.

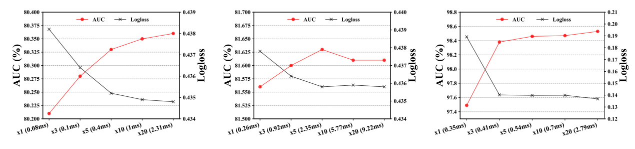
4.3.5 *Scalability Study.*

Scalability has always been one of the key concerns for deep learning models [27]. A good CTR model should flexibly adjust its capacity to adapt to different data patterns based on varying application scenarios. Theoretically, the



(a) ML-1M (b) Frappe (c) KKBox

Figure 6: Performance of different Loss in QNN-α.



(a) Tenrec (b) Criteo (c) Frappe

Figure 7: Performance of different M in QNN-α.

Table 5: Compatibility study of QNN-α.

Model	ML-1M		Frappe		iPinYou	
	Logloss ↓	AUC(%) ↑	Logloss ↓	AUC(%) ↑	Logloss ↓	AUC(%) ↑
DNN [9]	0.3100	90.30	0.1653	98.11	0.005545	78.06
DNN + SE Loss	0.3050	90.52	0.1616	98.25	0.005543	78.08
QNN-α	**0.2960**	**90.87**	**0.1313**	**98.62**	**0.005516**	**78.63**
xDeepFM [35]	0.3070	90.42	0.1405	98.45	0.005544	78.13
xDeepFM + SE Loss	0.3063	90.50	0.1363	98.52	0.005544	78.15
xDeepFM (QNN-α)	**0.2957**	**90.86**	**0.1329**	**98.61**	**0.005524**	**78.44**
DeepFM [16]	0.3073	90.51	0.1575	98.37	0.005549	77.94
DeepFM + SE Loss	0.3055	90.58	0.1504	98.53	0.005548	77.97
DeepFM (QNN-α)	**0.2966**	**90.82**	**0.1364**	**98.60**	**0.005533**	**78.27**
DCNv2 [67]	0.3070	90.42	0.1484	98.45	0.005539	78.26
DCNv2 + SE Loss	0.3037	90.53	0.1449	98.56	0.005536	78.20
DCNv2 (QNN-α)	**0.2957**	**90.91**	**0.1380**	**98.63**	**0.005527**	**78.49**

hyperparameter M in the KRP of Equation (9) effectively meets this requirement. To validate the scalability of KRP, we conduct experiments on two large-scale, highly sparse datasets and a small dataset. Figure 7 shows that as M increases, Logloss decreases and AUC increases on the Tenrec and Frappe datasets, demonstrating the scalability of QNN-α. Notably, when $M = 1$, the model performs worst as the KRP in Equation (9) degenerates into a simple HP. This validates the superiority of our proposed KRP.

4.3.6 *Compatibility Study.*

QNN-α can be viewed as a plug-and-play module designed to enhance the performance of deep CTR prediction models. To validate the compatibility of QNN-α, we replace the original MLP in four representative baseline models with QNN-α. The experimental results, shown in Table 5, indicate that the QNN-enhanced baseline models consistently achieve performance improvements and outperform their variants that only use SE Loss. This demonstrates the advantages of QNN in terms of both performance and compatibility. From an industrial application perspective, as increasingly complex ensemble models continue to emerge [41, 46], QNN-α can serve as a flexible component within sophisticated architectures, further boosting overall performance.

4.3.7 *Inference Efficiency Study.*

Inference efficiency is crucial in industrial recommender systems [81]. High inference latency can

Table 6: Inference efficiency study of different H in QNN-α.

Model		ML-1M			iPinYou			KKBox		
		Logloss	AUC(%)	Latency	Logloss	AUC(%)	Latency	Logloss	AUC(%)	Latency
FinalMLP [42]		0.3058	90.52	1.10ms	0.005556	78.02	0.78ms	0.4822	85.10	7.62ms
FINAL (2B) [80]		0.3035	90.53	0.68ms	0.005540	78.13	0.44ms	0.4795	85.08	8.65ms
QNN -α	x1 Head	0.2965	90.82	0.27ms	0.005536	78.13	0.79ms	**0.4726**	85.74	8.69ms
	x2 Head	**0.2960**	90.87	0.26ms	0.005529	78.22	0.51ms	0.4730	**85.76**	7.52ms
	x4 Head	**0.2960**	**90.88**	0.24ms	**0.005516**	**78.63**	0.23ms	0.4763	85.73	7.13ms
	x8 Head	0.2997	90.62	**0.18ms**	0.005529	78.28	**0.21ms**	0.4807	85.66	**5.75ms**

Table 7: Training efficiency study of QNN-α.

Model	Criteo				KKBox			
	Logloss	AUC(%)	Memory	Time	Logloss	AUC(%)	Memory	Time
xDeepFM	0.4376	81.43	15806MiB	781s	0.4772	85.35	11799MiB	109s
AutoInt*	0.4390	81.32	5158MiB	392s	0.4773	85.34	6449MiB	56s
w/o SE Loss	0.4363	81.57	**3996MiB**	**258s**	0.4783	85.57	**5795MiB**	**39s**
QNN-α	**0.4358**	**81.63**	5304MiB	328s	**0.4730**	**85.76**	8227MiB	63s

limit the practical use of accurate models. To evaluate QNN-α, we compare it with FinalMLP [42] and FINAL [80], which have been successfully applied in production environments. While KRP-based feature interaction increases computational cost, our multi-head KRP method alleviates this issue effectively. Experiments shown in Table 6 explore the impact of different head numbers (H) on performance and latency. Results show that increasing H reduces inference latency, confirming multi-head KRP's in optimizing inference efficiency. When $H = 4$, $H = 4$, and $H = 2$, QNN-α achieves optimal performance on ML-1M, iPinYou, and KKBox datasets, respectively. It surpasses FINAL in prediction accuracy and significantly reduces latency, demonstrating its strong performance and efficiency for real-world production environments.

4.3.8 ***Training Efficiency Study.*** Training efficiency is also a major concern for CTR researchers [32, 64, 81]. In real-world production environments, users generate data on the scale of hundreds of millions per day. Low training efficiency increases the additional costs of recommender systems. To evaluate QNN-α, we compare its training efficiency with xDeepFM [35] and AutoInt* [57]. The experimental results are shown in Table 7. We observe that QNN-α achieves GPU memory requirements and per-epoch training time similar to AutoInt*, while delivering significant performance improvements. Meanwhile, since SE Loss only requires two forward passes through QNN without additional operations on the embedding layer, its introduction does not result in a twofold decrease in training efficiency. In practice, researchers can decide whether to incorporate SE Loss according to specific requirements.

5 Related Work

5.1 CTR Prediction

Effectively capturing feature interactions is one of the key factors in improving the performance of CTR prediction models [6, 16, 66, 76]. Early CTR models are typically limited to capturing low-order feature interactions with bounded degrees [45, 53, 59, 70] or rely on hand-crafted feature combinations based on expert knowledge [6, 55]. With the rise of deep learning, some works attempt to leverage MLP to implicitly capture high-order feature interactions [16, 20, 50, 51, 66, 77], achieving promising results. However, research suggests that while MLP can theoretically serve as universal function approximators [23], they struggle to effectively model product-based feature interactions [54].

As a result, CTR researchers begin to explore paradigms that combine explicit and implicit feature interactions [35, 67, 68]. In recent years, increasingly sophisticated explicit feature interaction methods have been proposed [4, 7, 34, 37, 56, 57]. To evaluate these methods, CTR researchers establish open benchmarking systems [81], which reveal that, under well-tuned hyperparameter settings, the performance differences between models are often minimal. Consequently, some works attempt to break away from this conventional modeling paradigm and provide novel insights. Examples include only MLP-based [42], complex-space [61, 62], exponential [32, 80], contrastive learning-based [18, 31, 65], and LLM-enhanced [15, 49] feature interaction models.

5.2 Quadratic Neural Networks

QNN is a class of neural architectures designed to explicitly model second-order interactions between input features [8]. Unlike traditional MLP to learn implicitly, QNN leverages quadratic terms to capture feature interactions more directly and expressively [3, 10, 12]. Moreover, some studies demonstrate that QNN has stronger nonlinear approximation capabilities compared to MLP [26], and QNN can also be regarded as a universal function approximator [13]. QuadraLib [74] explores the optimization and design of QNN architectures. More recently, several works focus on further improving the computational efficiency of QNN [71, 73] and exploring higher-dimensional neural networks [72]. To the best of our knowledge, no existing work has attempted to think about the CTR prediction task from the perspective of QNN, even though classic models such as FM [53] and CrossNetv2 [67] can be directly interpreted as a QNN. Therefore, this work has the potential to provide new insights into CTR prediction.

6 Conclusion

In this paper, we revisited feature interaction models for CTR prediction from the QNN perspective. We found that the linearly independent quadratic polynomials in QNN were the core reason why HP played a significant role, as they expanded the feature space and introduced smooth nonlinear approximation capabilities. We evaluated 25 QNN neuron formats and observed that the traditional Post Act did not effectively enhance QNN's performance. We then proposed QNN-α, incorporating the Multi-Head Khatri-Rao Product as an improved HP and a Self-Ensemble Loss for dynamic ensemble predictions without extra sub-networks. Experiments show that QNN-α achieves a new SOTA performance on six public datasets while ensuring low latency, scalability, and compatibility.

Besides, it is noteworthy that we further integrate QNN with user behaviors sequence-based CTR prediction models and propose the Quadratic Interest Network [29]. This method achieves an impressive second place in the EReL@MIR [14] competition, further demonstrating the effectiveness and practical value of our method.

Acknowledgments

This work is supported by the National Science Foundation of China (No. 62272001 and No. 62206002).

References

[1] Linas Baltrunas, Karen Church, Alexandros Karatzoglou, and Nuria Oliver. 2015. Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild. *arXiv preprint arXiv:1505.03014* (2015).

[2] Bobbi Jo Broxson. 2006. The Kronecker Product. (2006).

[3] Jie Bu and Anuj Karpatne. 2021. Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM, 675–683.

[4] Jianxin Chang, Chenbin Zhang, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, and Kun Gai. 2023. PEPNet: Parameter and Embedding Personalized Network for Infusing with Personalized Prior Information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3795–3804.

[5] 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 *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3757–3766.

[6] 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.

[7] 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.

[8] Grigorios G Chrysos, Stylianos Moschoglou, Giorgos Bouritsas, Jiankang Deng, Yannis Panagakis, and Stefanos Zafeiriou. 2021. Deep polynomial neural networks. *IEEE transactions on pattern analysis and machine intelligence* 44, 8 (2021), 4021–4034.

[9] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 191–198.

[10] Nicholas DeClaris and Mu-chun Su. 1991. A novel class of neural networks with quadratic junctions. In *Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 1557–1562.

[11] Thomas G Dietterich et al. 2002. Ensemble learning. *The handbook of brain theory and neural networks* 2, 1 (2002), 110–125.

[12] Fenglei Fan, Wenxiang Cong, and Ge Wang. 2018. A new type of neurons for machine learning. *International journal for numerical methods in biomedical engineering* 34, 2 (2018), e2920.

[13] Fenglei Fan, Jinjun Xiong, and Ge Wang. 2020. Universal Approximation with Quadratic Deep Networks. *Neural Networks* 124 (2020), 383–392.

[14] Junchen Fu, Xuri Ge, Xin Xin, Haitao Yu, Yue Feng, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2025. The 1st EReL@MIR Workshop on Efficient Representation Learning for Multimodal Information Retrieval. arXiv:2504.14788 [cs.IR] https://arxiv.org/abs/2504.14788

[15] Zichuan Fu, Xiangyang Li, Chuhan Wu, Yichao Wang, Kuicai Dong, Xiangyu Zhao, Mengchen Zhao, Huifeng Guo, and Ruiming Tang. 2023. A unified framework for multi-domain ctr prediction via large language models. *ACM Transactions on Information Systems* (2023).

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

[17] Qiushan Guo, Xinjiang Wang, Yichao Wu, Zhipeng Yu, Ding Liang, Xiaolin Hu, and Ping Luo. 2020. Online Knowledge Distillation via Collaborative Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11020–11029.

[18] 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.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[20] 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.

[21] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[22] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.

[24] Huawei. 2021. An open-source CTR prediction library. https://fuxictr.github.io.

[25] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging Weights Leads to Wider Optima and Better Generalization. *arXiv preprint arXiv:1803.05407* (2018).

[26] Yiyang Jiang, Fan Yang, Hengliang Zhu, Dian Zhou, and Xuan Zeng. 2020. Nonlinear CNN: improving CNNs with quadratic convolutions. *Neural Computing and Applications* 32 (2020), 8507–8516.

[27] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020).

[28] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[29] Honghao Li, Hanwei Li, Jing Zhang, Yi Zhang, Ziniu Yu, Lei Sang, and Yiwen Zhang. 2025. Quadratic Interest Network for Multimodal Click-Through Rate Prediction. arXiv:2504.17699 [cs.IR] https://arxiv.org/abs/2504.17699

[30] Honghao Li, Lei Sang, Yi Zhang, Xuyun Zhang, and Yiwen Zhang. 2023. CETN: Contrast-enhanced Through Network for CTR Prediction. *arXiv preprint arXiv:2312.09715* (2023).

[31] Honghao Li, Lei Sang, Yi Zhang, and Yiwen Zhang. 2024. SimCEN: Simple Contrast-enhanced Network for CTR Prediction. In *Proceedings of the 32th ACM International Conference on Multimedia*.

[32] Honghao Li, Yiwen Zhang, Yi Zhang, Hanwei Li, Lei Sang, and Jieming Zhu. 2025. FCN: Fusing Exponential and Linear Cross Network for Click-Through Rate Prediction. arXiv:2407.13349 [cs.IR] https://arxiv.org/abs/2407.13349

[33] Honghao Li, Yiwen Zhang, Yi Zhang, and Lei Sang. 2024. Ensemble Learning via Knowledge Transfer for CTR Prediction. *arXiv preprint arXiv:2411.16122* (2024).

[34] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. FiGNN: Modeling feature interactions via graph neural networks for CTR prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 539–548.

[35] 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.

[36] Zhutian Lin, Junwei Pan, Shangyu Zhang, Ximei Wang, Xi Xiao, Shudong Huang, Lei Xiao, and Jie Jiang. 2024. Understanding the Ranking Loss for Recommendation with Sparse User Feedback. *arXiv preprint arXiv:2403.14144* (2024).

[37] 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.

[38] Shuangzhe Liu, Gotz Trenkler, et al. 2008. Hadamard, Khatri-Rao, Kronecker and other matrix products. *International Journal of Information and Systems Sciences* 4, 1 (2008), 160–177.

[39] Xu Ma, Xiyang Dai, Yue Bai, Yizhou Wang, and Yun Fu. 2024. Rewrite the Stars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5694–5703.

[40] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proc. icml*, Vol. 30. Atlanta, GA, 3.

[41] Siddarth Malreddy, Matthew Lawhon, Usha Amrutha Nookala, Aditya Mantha, and Dhruvil Deven Badani. 2024. Improving feature interactions at Pinterest under industry constraints. *arXiv preprint arXiv:2412.01985* (2024).

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

[43] Diganta Misra. 2019. Mish: A Self-regularized Non-monotonic Activation Function. *arXiv preprint arXiv:1908.08681* (2019).

[44] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When Does Label Smoothing Help? *Advances in neural information processing systems* 32 (2019).

[45] 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.

[46] Junwei Pan, Wei Xue, Ximei Wang, Haibin Yu, Xun Liu, Shijie Quan, Xueming Qiu, Dapeng Liu, Lei Xiao, and Jie Jiang. 2024. Ads Recommendation in A Collapsed and Entangled World. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5566–5577.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019).

[48] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine Learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[49] Zexuan Qiu, Jieming Zhu, Yankai Chen, Guohao Cai, Weiwen Liu, Zhenhua Dong, and Irwin King. 2024. EASE: Learning Lightweight Semantic Feature Adapters

from Large Language Models for CTR Prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4819–4827.

[50] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based neural networks for user response prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1149–1154.

[51] 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.

[52] Prajit Ramachandran, Barret Zoph, and Quoc V Le. 2017. Searching for activation functions. *arXiv preprint arXiv:1710.05941* (2017).

[53] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International Conference on Data Mining*. IEEE, 995–1000.

[54] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 240–248.

[55] 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.

[56] Lei Sang, Honghao Li, Yiwen Zhang, Yi Zhang, and Yun Yang. 2024. AdaGIN: Adaptive Graph Interaction Network for Click-Through Rate Prediction. *ACM Transactions on Information Systems* (2024).

[57] 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.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[59] 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.

[60] Antti Tarvainen and Harri Valpola. 2017. Mean Teachers Are Better Role Models: Weight-averaged Consistency Targets Improve Semi-supervised Deep Learning Results. *Advances in neural information processing systems* 30 (2017).

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

[62] Zhen Tian, Yuhong Shi, Xiangkun Wu, Wayne Xin Zhao, and Ji-Rong Wen. 2024. Rotative Factorization Machines. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2912–2923.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* 30 (2017).

[64] Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Towards Deeper, Lighter and Interpretable Cross Network for CTR Prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2523–2533.

[65] 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*. 805–813.

[66] 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.

[67] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. DCNv2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*. 1785–1797.

[68] 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).

[69] Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, Tie-Yan Liu, et al. 2021. R-drop: Regularized Dropout for Neural Networks. *Advances in Neural Information Processing Systems* 34 (2021), 10890–10905.

[70] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: learning the weight of feature interactions via attention networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 3119–3125.

[71] Chenhui Xu, Xinyao Wang, Fuxun Yu, JInjun Xiong, and Xiang Chen. 2024. QuadraNet V2: Efficient and Sustainable Training of High-Order Neural Networks with Quadratic Adaptation. *arXiv preprint arXiv:2405.03192* (2024).

[72] Chenhui Xu, Fuxun Yu, Maoliang Li, Zihao Zheng, Zirui Xu, Jinjun Xiong, and Xiang Chen. 2024. Infinite-Dimensional Feature Interaction. *arXiv preprint arXiv:2405.13972* (2024).

[73] Chenhui Xu, Fuxun Yu, Zirui Xu, Chenchen Liu, Jinjun Xiong, and Xiang Chen. 2024. QuadraNet: Improving High-Order Neural Interaction Efficiency with Hardware-Aware Quadratic Neural Networks. In *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 19–25.

[74] Zirui Xu, Fuxun Yu, Jinjun Xiong, and Xiang Chen. 2022. Quadralib: A Performant Quadratic Neural Network Library for Architecture Optimization and Design Exploration. *Proceedings of Machine Learning and Systems* 4 (2022), 503–514.

[75] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. 2022. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. *Advances in Neural Information Processing Systems* 35 (2022), 11480–11493.

[76] Kexin Zhang, Fuyuan Lyu, Xing Tang, Dugang Liu, Chen Ma, Kaize Ding, Xiuqiang He, and Xue Liu. 2025. Fusion Matters: Learning Fusion in Deep Click-through Rate Prediction Models. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25)*. 744–753.

[77] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1059–1068.

[78] Chenxu Zhu, Peng Du, Weinan Zhang, Yong Yu, and Yang Cao. 2022. Combo-fashion: Fashion clothes matching CTR prediction with item history. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4621–4629.

[79] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. Bars: Towards open benchmarking for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2912–2923.

[80] Jieming Zhu, Qinglin Jia, Guohao Cai, Quanyu Dai, Jingjie Li, Zhenhua Dong, Ruiming Tang, and Rui Zhang. 2023. FINAL: Factorized interaction layer for CTR prediction. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2006–2010.

[81] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-through Rate Prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2759–2769.