UniMixer: A Unified Architecture for Scaling Laws in Recommendation Systems

Mingming Ha **Guanchen Wang** **Linxun Chen** **Xuan Rao** **Yuexin Shi** **Tianbao Ma**
Zhaojie Liu **Yunqian Fan** **Zilong Lu** **Yanan Niu** **Han Li** **Kun Gai**
Kuaishou Technology, Beijing, China
{hamingming, wangguanchen, chenxi36, raoxuan, shiyuexin, matianbao,
zhaotianxing, fanyunqian03, luzilong, niuyanan, lihan08}@kuaishou.com, gaikun@qq.com

Abstract

In recent years, the scaling laws of recommendation models have attracted increasing attention, which govern the relationship between performance and parameters/FLOPs of recommenders. Currently, there are three mainstream architectures for achieving scaling in recommendation models, namely attention-based, TokenMixer-based, and factorization-machine-based methods, which exhibit fundamental differences in both design philosophy and architectural structure. In this paper, we propose a unified scaling architecture for recommendation systems, namely **UniMixer**, to improve scaling efficiency and establish a unified theoretical framework that unifies the mainstream scaling blocks. By transforming the rule-based TokenMixer to an equivalent parameterized structure, we construct a generalized parameterized feature mixing module that allows the token mixing patterns to be optimized and learned during model training. Meanwhile, the generalized parameterized token mixing removes the constraint in TokenMixer that requires the number of heads to be equal to the number of tokens. Furthermore, we establish a unified scaling module design framework for recommender systems, which bridges the connections among attention-based, TokenMixer-based, and factorization-machine-based methods. To further boost scaling ROI, a lightweight UniMixing module is designed, **UniMixing-Lite**, which further compresses the model parameters and computational cost while significantly improve the model performance. The scaling curves are shown in the following figure. Extensive offline and online experiments are conducted to verify the superior scaling abilities of **UniMixer**.

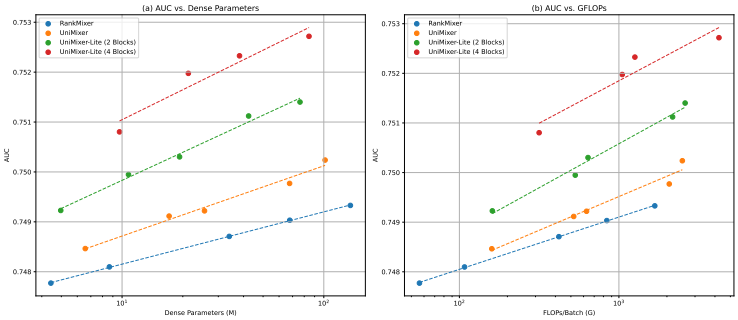

Figure 1: The scaling laws between AUC and Parameters for the present UniMixer/UniMixer-Lite and RankMixer architectures. The x-axis is presented on a logarithmic scale.

1 Introduction

Large language models (LLMs) have revealed an impressive phenomenon: as model size, data volume, and computational resources increase, performance improves steadily, namely scaling laws. The remarkable performance scaling observed in LLMs has inspired the recommender systems community to explore scaling frameworks tailored to recommendation tasks. In recent years, researchers have attempted to design scaling modules and stack them across multiple layers to increase the complexity of ranking models, thereby achieving scaling laws between model performance and model size or computational cost (e.g., parameters and FLOPs).

Based on a large amount of multi-field user and item feature, recommender systems predict the user behaviors to present the most relevant content to them to increase user's positive engagements with the recommendations. These multi-field features generally involve categorical features and dense features, which generally possess more dynamic embedding representations and capture information from multiple perspectives. Differing from the natural language processing (NLP) domain, where all tokens share a unified embedding space, the feature space in recommendation tasks is inherently heterogeneous. Therefore, learning heterogeneous features interactions represents a fundamental difference from NLP domain. Owing to the tremendous success of Transformers in LLMs, a natural idea is to modify the Transformer module to adapt recommendation tasks because it is generally infeasible to directly adopt Transformer module as the fundamental block for scaling laws in recommendation systems. To address the heterogeneous feature interaction problem, current mainstream scaling architectures for recommendation models can be broadly categorized into three types: attention-based, TokenMixer-based, and factorization-machine-based methods. Attention-based methods (e.g., HiFormer [1], FAT [2], and HHFT [3], etc.) construct token-specific query, key, and value projections for each input token. In contrast to attention-based methods, TokenMixer-based methods (e.g., RankMixer [4], TokenMixer-Large [5], etc.) employ the rule-based token mixing operation to achieve heterogeneous feature interactions, which avoids computing inner-product similarity between two heterogeneous semantic spaces. Factorization-machine-based methods (e.g. Wukong [6], Kunlun [7], etc.), on the other hand, model feature interactions by introducing a Factorization Machine (FM) block to compute interactions among the input embeddings within each layer. These frameworks are built upon completely different scaling blocks, yet all demonstrate the capability to scale up model performance. This leads us to a fundamental question: Can we construct a unified scaling module for recommendation systems that combines the advantages of existing mainstream scaling components? To bridge the connections among these scaling modules, we first find a parameterized formulation of the rule-based TokenMixer operation. By further optimizing the computation pipeline, we derive the UniMixing module with reduced computational cost. Based on this design and results, we propose a unified theoretical framework that unifies the mainstream scaling modules in recommender systems. Besides, a lightweight UniMixer module is designed, which use the advantages of existing mainstream scaling blocks and achieve the best parameter efficiency and computational efficiency. We hope that the unified architecture can help the recommendation systems community achieve its own "attention moment". Our main contributions can be summarized as follows:

1) We reveal the feature interaction patterns of TokenMixer via equivalent parameterization of the rule-based TokenMixer.

2) We propose a unified scaling framework, termed UniMixer, which bridges the differences and connections among attention-based, TokenMixer-based, and FM-based methods. By optimizing the computation pipeline, UniMixer significantly reduces computational complexity and GPU memory consumption during both training and inference.

3) To further reduce model parameters and computational cost, we design a lightweight UniMixing module, called UniMixing-Lite, which can simultaneously leverage the advantages of both attention-based and TokenMixer-based architecture to achieve improved scaling efficiency.

4) Extensive offline and online experiments are performed to demonstrate the superior scaling abilities of UniMixer.

2 Related Work

Currently, there are three scaling modeling paradigms for establishing scaling laws for massive-scale recommendation systems: attention-based, TokenMixer-based,and FM-based methods.

Attention-Based Framework. Recent research in recommendation systems has adapted Transformers for CTR prediction. A core challenge in this paradigm is bridging the gap between the heterogeneous nature of the token sequence and the sequential compositionality that assumed by language modeling. To this end, in [1], the heterogeneous attention layer is proposed to address the heterogeneous feature interaction and HiFormer is designed to explicitly model high-order interactions by flattening heterogeneous tokens into a single vector representation. Additionally, Field-Aware Transformers (FAT) inject field-aware interaction priors into the attention mechanism via factorized contextual alignment and cross-field modulation [2], further establishing the empirical scaling law for CTR prediction. HHFT further validates these scaling properties by interleaving heterogeneous Transformer blocks (for preserving domain-specific semantics) with HiFormer blocks (for high-order interaction learning) [3]. Furthermore, in dynamic user behavior modeling, methods such as HSTU-V1/V2 [8, 9], MARM [10], OneTrans [11], Climber [12], Hyformer [13], and LLaTTE [14] leverage attention mechanisms to capture long-range temporal dependencies. These approaches highlight the potential of unifying feature interaction and sequential behavior modeling to achieve more robust scaling laws.

TokenMixer-Based Framework. While attention mechanisms offer expressive feature interaction, they incur prohibitive computational costs due to the quadratic complexity of attention score computation. Inspired by the success of MLP-Mixer [15] in computer vision, a paradigm shift towards token-mixing architectures has emerged in industrial recommender systems, yielding advanced models such as RankMixer [4], Lemur [16], and TokenMixer-Large [5]. For instance, RankMixer replaces dynamic attention with static, non-parametric token-mixing operations, achieving competitive CTR prediction performance while maintaining strictly comparable FLOPs [4]. Building upon this, TokenMixer-Large scales this architecture to 13B configurations by introducing auxiliary residual connections and tailored loss functions [5], demonstrating compelling scaling laws across various model dimensions. Nonetheless, a critical gap remains: the design of current token-mixing operators heavily relies on empirical rules and lacks a rigorous theoretical bridge to traditional FM-based or attention-based methodologies.

FM-Based Framework. The pioneer FM-based method employs the low-order pairwise modeling for feature interactions in recommendation systems [17], which was subsequently generalized by Field-aware FMs to capture field-specific and context-sensitive interactions [18]. While these models benefit from high interpretability and efficiency, they are constrained intrinsically by their capacity of low-order interactions. To address this limitation, various neural network-based extentions, such as DeepFM [19], AutoInt [20], and DCN series [21, 22], integrate MLP or transformer attention to capture high-order interactions. More recently, Wukong [6] has demonstrated appropriate scaling properties by stacking FM-style interaction blocks with linear compression. Nevertheless, the reliance on explicit low-order interaction of FM-based methods still limits the performance improvement when models are scaled up in terms of parameters and FLOPs, which is in contrast to the predictive scaling laws observed in LLMs [23, 24].

3 Preliminaries

Consider a class of discriminative recommendation tasks, such as rating, click-through rate (CTR) and post-click conversion rate (CVR) predictions, and so forth, which are typically formulated as a supervised learning problem. The dataset is defined as $\mathcal{D} = \{(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_i, y_i), \ldots, (\mathbf{X}_N, y_N)\}$, where $\mathbf{X}_i = \left[\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(2)}, \ldots, \mathbf{x}_i^{(F)}\right]$ with F feature fields, $y_i \in \{0, 1\}$ corresponding to a binary classification problem or $y_i \in \mathbb{R}$ for a regression problem is the label of the i-th sample, N is the number of data points. In general, the input features $\mathbf{X} = \{\mathbf{X}^{\mathrm{C}}, \mathbf{X}^{\mathrm{D}}\}$ are divided into categorical features $\mathbf{X}^{\mathrm{C}}$ and dense features $\mathbf{X}^{\mathrm{D}}$. $|C|$ and $|D|$ are used to denote the numbers of categorical and dense features, respectively. For CTR and CVR prediction tasks, the core objective is to establish a model to predict the click or conversion probability $\Pr(y_i = 1|\mathbf{X}_i)$. In recommender systems, the learned embedding representations are more dynamic [1]. Differing from input tokens of language

models, the feature spaces are inherently heterogeneous [4]. Therefore, it is inappropriate to directly transfer the Transformer architecture used in large language models to recommendation modeling. To date, scaling laws in the recommendation domain have primarily been established through three types of foundational blocks and their variants.

Heterogeneous Attention Layer. Heterogeneous-attention-based architecture [1, 2, 3] generally use the field-specific query, key, and value projections to achieve heterogeneous feature interaction. Given an input hidden states $X = [\boldsymbol{x}_1; \dots ; \boldsymbol{x}_T] \in \mathbb{R}^{T \times D}$, the heterogeneous attention layer is formulated as

$$Q_h = \begin{bmatrix} \boldsymbol{x}_1 W_Q^{1h} \\ \vdots \\ \boldsymbol{x}_T W_Q^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, K_h = \begin{bmatrix} \boldsymbol{x}_1 W_K^{1h} \\ \vdots \\ \boldsymbol{x}_T W_K^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, V_h = \begin{bmatrix} \boldsymbol{x}_1 W_V^{1h} \\ \vdots \\ \boldsymbol{x}_T W_V^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, \quad (1)$$

where $W_Q^{ih}, W_K^{ih}, W_V^{ih} \in \mathbb{R}^{D \times d}$ are the token-specific weights of query, key, and value projections, respectively. The output of the multi-head heterogeneous attention layer is computed as follows

$$O_h = \text{softmax}\Big(\frac{Q_h K_h^{\mathsf{T}}}{\sqrt{d}}\Big) V_h \in \mathbb{R}^{T \times d}. \quad (2)$$

Then the outputs of the multi-head heterogeneous attention are concatenated and passed through a linear projection to align the output dimension with the input X.

TokenMixer. TokenMixer-based framework [4, 5, 25] employ the parameter-free and rule-based mixing operation to perform the feature interaction. For the given input $X = [\boldsymbol{x}_1; \dots ; \boldsymbol{x}_T]$, TokenMixer first evenly splits each input token $\boldsymbol{x}_t$ into H heads.

$$\Big[\boldsymbol{x}_t^{(1)} | \, \boldsymbol{x}_t^{(2)} \mid \dots \mid \boldsymbol{x}_t^{(H)}\Big] = \text{SplitHead}(\boldsymbol{x}_t). \quad (3)$$

Then, the h-token $\boldsymbol{s}^h$ can be obtained as

$$\boldsymbol{s}^h = \text{concat}\big(\boldsymbol{x}_1^{(h)}, \boldsymbol{x}_2^{(h)}, \dots, \boldsymbol{x}_T^{(h)}\big) \in \mathbb{R}^{\frac{TD}{H}} \quad (4)$$

The output of TokenMixer is formulated as

$$S = \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_H \end{bmatrix} \in \mathbb{R}^{H \times \frac{TD}{H}}, \quad (5)$$

where H is required to be the same as T. Therefore, dimensions of the input X and the output S are identical.

Wukong. Wukong-based models [6, 7]concatenate the outputs of a Factorization Machine Block (FMB) and a linear projection layer to upscale the interaction component.

$$\begin{aligned} &\text{FMB}(X) = \ \text{reshape}(\text{MLP}(\text{LN}(\text{flatten}(\text{FM}(X))))), \ \text{FM}(X) = XX^{\mathsf{T}}Y \\ &\text{LCB}(X) = \ WX \end{aligned} \quad (6)$$

where $W \in \mathbb{R}^{n \times T}$ and $Y \in \mathbb{R}^{T \times r}$ are learnable projection matrix. Y is used to reduce memory requirement to store the interaction matrix XX^{T}.

In this work, we focus on establishing a unified structural foundation for recommendation systems that integrates the strengths of current scaling blocks to further increase the scaling ROI.

4 UniMixer

4.1 Overview

A unified module, namely the UniMixer block, for scaling up recommender systems is established, which unifies the mainstream scaling modules for recommendation such as attention-based modules,

TokenMixer-based modules, and Wukong-based methods, under a unified theoretical framework. As shown in Fig. 2, the overall architecture consists of feature tokenization, M UniMixer blocks with the Siamese norm and Sparse-Pertoken MoE. Through parameterized rule-based TokenMixer, we bridge the connection among attention-based, TokenMixer-based, and Wokong-based methods, enabling the proposed UniMixer to simultaneously possess the advantages of these approaches. Besides, a lightweight UniMixing module is developed to further compresses the model parameters and computational cost while significantly improve the model performance.

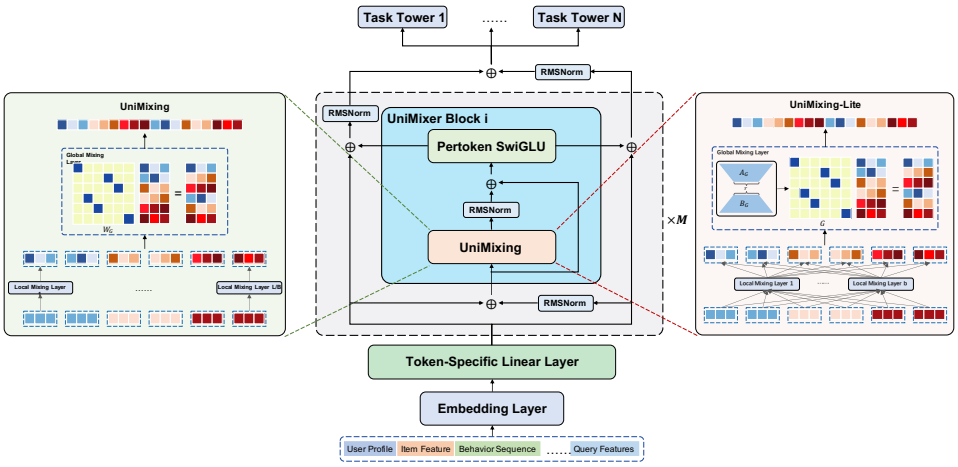

Figure 2: The UniMixer architecture for scaling laws in recommendation systems.

4.2 Feature Tokenization

Based on the semantic categories of the input feature fields, the input features $\mathbf{X}$ is first divided into N disjoint feature domains

$$\mathbf{X} = \Big[\underbrace{\mathbf{x}_U^{(1)}, \ldots, \mathbf{x}_U^{(n_U)}}_{\text{User Profile}}, \underbrace{\mathbf{x}_I^{(1)}, \ldots, \mathbf{x}_I^{(n_I)}}_{\text{Item Features}}, \underbrace{\mathbf{x}_B^{(1)}, \ldots, \mathbf{x}_B^{(n_B)}}_{\text{Behavior Sequence}}, \underbrace{\mathbf{x}_Q^{(1)}, \ldots, \mathbf{x}_Q^{(n_Q)}}_{\text{Query Features}}, \ldots\Big]. \tag{7}$$

Each feature domain is transformed into different embedding vectors with dimension by embedding layers

$$\mathbf{e}_n = \text{Embedding}(\mathbf{X}_{\text{domain}}) \in \mathbb{R}^{d_{\text{domain}}}, \tag{8}$$

where $\mathbf{X}_{\text{domain}}$ denotes all the features within a feature domain, d_{domain} is the embedding dimension corresponding this feature domain. The obtained feature domain embeddings are concatenated into one embedding vector $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]$. Similar to [4], the embedding vector $\mathbf{E}$ is evenly divided into an appropriate number of block. Then, each block is projected into a token embedding by using the following token-specific linear layer

$$\boldsymbol{x}_i = W_i^{\text{proj}} \mathbf{E}_{di:di+d} + \mathbf{b}_i^{\text{proj}} \in \mathbb{R}^D, \tag{9}$$

where $W_i^{\text{proj}} \in \mathbb{R}^{D \times d}$, $\mathbf{b}_i^{\text{proj}} \in \mathbb{R}^D$. The input hidden states $X \in \mathbb{R}^{T \times D}$ can then be obtained by stacking $\boldsymbol{x}_i$ column-wise.

4.3 UniMixer Block

Heterogeneous Feature Interactions. As mentioned in Section 3, heterogeneous attention addresses the feature interaction problem between two heterogeneous semantic spaces by employing token-specific query, key, and value weights. However, the attention pattern obtained by computing the inner-product similarity typically carries a diagonally dominant prior. In the early stage of training, with randomly initialized weights W_Q^h and W_K^h, the magnitude of the attention weights is largely dominated by the input token values X, which can easily cause the attention weights to concentrate on a small number of tokens, as shown in Fig. 3(a).

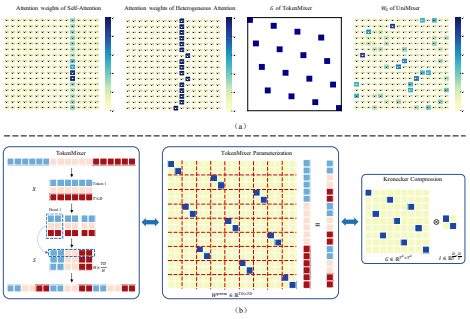

Figure 3: (a) The global mixing weights of different methods. (b) Equivalent parameterization of the rule-based TokenMixer.

As illustrated in the Fig. 3(a), it can be observed that the attention weights of the heterogeneous attention are sharp and sparse, which pose a risk to gradient backpropagation, thereby making the training of the query and key weights difficult and potentially causing it to stall, as shown in Fig. 3(a) (the 10-th and 15-th row in attention weights of the heterogeneous attention). Meanwhile, under large-scale heterogeneous feature inputs, such attention patterns may lead to uniform feature interactions, namely, attention scores become very small and lack discriminability, which potentially result in noise signals to obscure critical feature interaction patterns.

On the other hand, the parameter-free and rule-based TokenMixer operation lacks learnability and scenario adaptability, which can lead to insufficient or erroneous heterogeneous feature interactions. In addition, requiring $T = H$ further restricts the selection of heterogeneous feature interaction patterns. Through in-depth analysis of the TokenMixer operation, we have made some interesting findings, which make it possible to parameterize the TokenMixer operation. As shown in Figure 3(b), we observe that the TokenMixer operation can be regarded as the product of a permutation matrix W^{perm} and the flattened input embedding $\text{flatten}(X) \in \mathbb{R}^{TD}$, which can be formulated as

$$\text{TokenMixer}(X) = \text{reshape}(W^{\text{perm}}\text{flatten}(X)), \tag{10}$$

where $W^{\text{perm}} \in \mathbb{R}^{TD \times TD}$ is a large permutation matrix. A concrete numerical example is provided in Appendix A. A natural idea is to enable rule-based TokenMixer to be learnable and optimizable by parameterizing the permutation matrix W^{perm}. However, the computation complexity $O(T^2D^2)$ and the number of parameters $O(T^2D^2)$ is unacceptable. Through observation, we have made some interesting findings regarding the permutation matrix W^{perm} of TokenMixer and summarize them as the insightful properties in the following box.

Properties of The Permutation Matrix W^{perm}.

- **Compressibility.** The permutation matrix W^{perm} can be equivalently decomposed into the Kronecker product of two smaller matrices, i.e., $W^{\text{perm}} = G \otimes I$, where $I \in \mathbb{R}^{\frac{D}{T} \times \frac{D}{T}}$ is the identity matrix and $G \in \mathbb{R}^{T^2 \times T^2}$, $\otimes$ is the Kronecker product operation.
- **Doubly Stochasticity.** In the permutation matrix W^{perm}, every row and every column sums to 1, i.e., $\sum_{p=1}^{TD} w_{pq}^{\text{perm}} = 1, \sum_{q=1}^{TD} w_{pq}^{\text{perm}} = 1$.
- **Sparsity.** Each row/column of W^{perm} contains exactly one non-zero element.
- **Symmetry.** If $T = H$, the permutation matrix W^{perm} is a symmetric matrix, i.e., $W^{\text{perm}} = W^{\text{perm}\mathsf{T}}$. If $T \neq H$, the permutation matrix W^{perm} is a asymmetric matrix.

According to the properties of the permutation matrix of TokenMixer, the number of parameters for token mixing is significantly reduced by parameterizing the matrices G and I, namely, $O(T^4 + (\frac{D}{T})^2)$, where T is typically much smaller than D. Besides, there remain three challenges in the parameterization of the TokenMixer: (1) Directly using parameterized G and I to reconstruct W^{perm} still produces an intermediate variable of size $[TD, TD]$ during model training and inferring processes, which imposes a very high demand on GPU memory; (2) How to ensure that the learned parameters satisfy doubly stochasticity, sparsity and symmetry; (3) How to design a unified recommendation scaling module that integrates the strengths of existing scaling modules and establish superior scaling efficiency for the recommendation systems.

Unified Token Mixing Module. Inspired by Fig. 3, in the unified token mixing module, T and D are no longer used; instead, we define the block and the block size in the permutation matrix. The block size is denoted as B. The number of blocks is $(L//B)^2$, where L is the input embedding dimension and can be divisible by the block size B. Denote the parameterized weights of G as

W_G. Considering the sparsity of the permutation matrix and to achieve sufficient heterogeneous feature interactions, we assign a distinct parameterized weight W_B^i to each row. With this operation, each block possesses a different feature interaction pattern. Then, a permutation matrix W^{perm} with richer interaction patterns can be obtained by learning the parameter matrices W_G and W_B^i, which is formulated as

$$\text{UniMixing}(X) = \text{reshape}\Big(\Big(W_G \otimes \{W_B^i\}_{i=1}^{L//B}\Big)\text{flatten}(X), 1, L\Big), \tag{11}$$

where $\otimes$ is the generalized Kronecker product.

Next, the computation pipeline of (11) is optimized to significantly reduce both the computational cost and GPU memory requirements. The embedding vector flatten(X) is first evenly split into $L//B$ vectors and the size of each vector is B, which is expressed as

$$\Big[\boldsymbol{x}_1 |\ \boldsymbol{x}_2 \mid \ \dots \ \mid \boldsymbol{x}_{\frac{L}{B}}\Big] = \text{Split}\Big(\text{flatten}(X), \frac{L}{B}\Big). \tag{12}$$

Then, the block weights W_B^i are multiplied with the corresponding block-wise vectors $\boldsymbol{x}^{(i)}$, respectively, to obtain the following local feature interaction vector

$$\text{reshape}\Big(H, \frac{L}{B}, B\Big) = \text{reshape}\Big(\Big[\boldsymbol{x}_1 W_B^1 \ \Big|\ \boldsymbol{x}_2 W_B^2 \ \Big|\ \dots \ \Big|\ \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}}\Big], \frac{L}{B}, B\Big) = \begin{bmatrix} \boldsymbol{x}_1 W_B^1 \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}} \end{bmatrix}. \tag{13}$$

Therefor, the output of UniMixing module can be obtained as

$$\text{UniMixing}(X) = \text{reshape}\Big(W_G \text{reshape}\Big(H, \frac{L}{B}, B\Big), 1, L\Big). \tag{14}$$

With this operation, compared to directly using the reconstructed matrix W^{perm}, the optimized computation pipeline reduces the computational cost from $O(L^2)$ to $O(L^2/B + LB)$, and avoids the creation of large intermediate variables during computation. The proof of the computation pipeline optimization of (11) is provided in Appendix B. According to (13) and (14), W_B^i controls the intra-block interaction pattern, while W_G controls the inter-block interaction pattern. For the embedding inputs with dimension L, it is no longer required that $T = H$. Compared with the TokenMixer operation, the UniMixing module possesses more diverse local and global feature mixing patterns and interaction scales, while also retaining the advantage of being learnable and optimizable. To fulfill the doubly stochasticity of learned permutation matrices, Sinkhorn-Knopp iteration is used to makes all elements of W_G and W_B^i to be positive via an exponent operator and then conducts iterative normalization that alternately rescales rows and columns to sum to 1. Besides, a temperature coefficient is introduced to control the sparsity of the parameter matrix. Finally, we employ $(W_G + W_G^{\mathsf{T}})/2$ and $(W_B^i + W_B^{i\mathsf{T}})/2$ to achieve the symmetry constraints of parameter matrices. The final constrained weights can be obtained by

$$\begin{aligned} &\tilde{W}_G = \frac{W_G + W_G^{\mathsf{T}}}{2},\ \tilde{W}_B^i = \frac{W_B^i + W_B^{i\mathsf{T}}}{2}, \\ &\bar{W}_G = \text{Sinkhorn-Knopp}\Big(\frac{\tilde{W}_G}{\tau}\Big), \bar{W}_B^i = \text{Sinkhorn-Knopp}\Big(\frac{\tilde{W}_B^i}{\tau}\Big), \end{aligned} \tag{15}$$

where τ is the temperature coefficient.

Then, the residual connection and normalization module are used to process the output of the UniMixing block

$$O = \text{RMSNorm}(X + \text{UniMixing}(X)) \tag{16}$$

A Unified Perspective of Heterogeneous Feature Interaction. Observing V_h in (1) and reshape$(B, L/B, B)$ in (13), we find that if the number of blocks $L//B$ is set as T, and W_V^{ih} and W_B^i have the same dimensions, then reshape$(B, L/B, B) = V_h$. This implies that the local interaction projection of UniMixer is equivalent to the value projection of the heterogeneous attention layer under $W_V^i = W_B^i$. On the other hand, the dimension and the role of W_G are the same as the attention weights, except that W_G needs to satisfy the doubly stochasticity, sparsity, and symmetry.

Table 1: The differences of attention-based, TokenMixer-based, FM-based methods under the unified theoretical framework.

Heterogeneous Feature Mixing	Local Mixing Pattern	Global Mixing Pattern $G(X, W_G)$
Self-Attention	XW_V	$\text{softmax}(\frac{(XW_Q)(XW_K)^\intercal}{\sqrt{d}})$
Heterogeneous Attention	$X\tilde{W}_V$	$\text{softmax}(\frac{(X\tilde{W}_Q)(X\tilde{W}_K)^\intercal}{\sqrt{d}})$
TokenMixer	X	G
FM	Y	$XI(XI)^\intercal$

The feature interaction of Wukong is based on the FM component. Accoriding to (6), the expression of $\text{FMB}(X)$ can be rewritten as $\text{FMB}(X) = \text{reshape}(\text{MLP}(\text{LN}(\text{flatten}(XI(XI)^\intercal Y))))$, where I is the identity matrix with an appropriate dimension. Let us focus on the core feature interaction module $XI(XI)^\intercal Y$. In the attention module, when $W_Q = I$, $W_K = I$, and the value matrix does not depend on the hidden state input X, namely, $V_h = W_V = Y$, the Attention mechanism degenerates into the FM module. Therefore, attention-based, TokenMixer-based, and Wukong-based architecture can be unified under the following single theoretical framework

$$\text{UniMixing}(X) = \text{reshape}\Bigg(\underbrace{G(X, W_G)}_{\text{Global Mixing Pattern}} \underbrace{\begin{bmatrix} \boldsymbol{x}_1 W_B^1 \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}} \end{bmatrix}}_{\text{Local Mixing Pattern}}, 1, L\Bigg), \tag{17}$$

where $G(X, W_G)$ is a heterogeneous feature interaction projection and measures token-to-token/block-to-block interaction strength. To facilitate the analysis of the differences and connections among various methods, we consider the single-head attention setting. Under the unified theoretical framework (17), their differences are summarized in Table 1. For the self-attention, heterogeneous attention, and FM, the global mixing pattern $G(X, W_G)$ is obtained by computing the inner-product similarity between two tokens. The global mixing pattern of TokenMixer is independent of the input token embedding.

UniMixing-Lite. As shown in Fig. 3, it can be observed that as the block granularity becomes finer, the number of local interaction parameter matrices W_B^i increases, and the global interaction parameter matrix W_G becomes larger. This leads to redundant local interaction patterns. Meanwhile, the larger global interaction matrix is not efficient in reducing the number of parameters. Therefore, based on the UniMixing block, we design a lightweight UniMixing module, UniMixing-Lite, to further reduce the number of module parameters and computational cost, thereby improving the scaling efficiency of the model.

To address the problem of redundancy in the local interaction pattern, a basis-composed module is introduced to dynamically generates the block-specific local mixing weight. Define a set of basis matrices for W_B^i as $\{Z_\ell\}_{\ell=1}^b$ and block-specific weight vectors over these bases as $\{\boldsymbol{\omega}^i\}_{i=1}^{L//B}$, where b is the number of the basis local mixing weight and $\boldsymbol{\omega}^i = [\omega_1^i, \ldots, \omega_b^i]$. In addition, for the global interaction parameter W_G, we use low-rank approximation to further improve the efficiency. Then, the UniMixing-Lite module can be expressed as

$$\begin{aligned} \text{UniMixing-Lite}(X) &= \text{reshape}\Big(W_r \text{reshape}\Big(\Big[\boldsymbol{x}_1 W_B^{*1} \;\Big|\; \ldots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W_B^{*\frac{L}{B}}\Big], \frac{L}{B}, B\Big), 1, L\Big), \\ O &= \text{RMSNorm}(X + \text{UniMixing-Lite}(X)), \end{aligned} \tag{18}$$

where $W_r = \text{Sinkhorn-Knopp}(A_G B_G)$, $W_B^{*i} = \text{Sinkhorn-Knopp}(\sum_{\ell=1}^b \omega_\ell^i Z_\ell)$, $A_G \in \mathbb{R}^{(L//B)\times r}$ and $B_G \in \mathbb{R}^{r\times(L//B)}$. r is the rank of the low-rank approximation for W_G. In the UniMixing-Lite module, we retain both the low-parameterized global interaction pattern of the TokenMixer and the local interaction capability of attention for heterogeneous features. It can simultaneously leverage the advantages of both attention-based and token-mixer-based methods.

Pertoken SwiGLU. After the UniMixing block, similar to [5], the pertoken SwiGLU is introduced to model the feature heterogeneity among different tokens. For each input token $\boldsymbol{x}_i$, the SwiGLU

formulation is given as follows

$$\text{pSwiGLU}(\boldsymbol{o}_i) = W^i_{\text{down}}((W^i_{\text{up}}\boldsymbol{o}_i + b^i_{\text{up}}) \odot \text{Swish}(W^i_{\text{gate}}\boldsymbol{o}_i + b^i_{\text{gate}})) + b^i_{\text{down}}, \tag{19}$$

where $W^i_{\text{up}}, W^i_{\text{gate}} \in \mathbb{R}^{B\times nB}$, $W^i_{\text{down}} \in \mathbb{R}^{nB\times B}$, $b^i_{\text{up}}, b^i_{\text{gate}} \in \mathbb{R}^{nB}$, $b^i_{\text{down}} \in \mathbb{R}^B$, $\boldsymbol{o}_i$ is the UniMixing output of the i-th token, and n is a hyperparameter.

4.4 SiameseNorm

The current RankMixer architecture [4] lacks specialized design for deep architectures, which is generally reflected in the limited effectiveness of scaling along the model depth. Although TokenMixer-Large [5] attempts to address this problem by incorporating interval residuals and the auxiliary loss within the TokenMixer-Large Block, it does not address the root of the problem. To achieve the training stability and performance gains as model depth increases, SiameseNorm [26] is introduced into the UniMixer architecture as shown in Fig. 2. As mentioned in [26], SiameseNorm resolves the tension between Pre-Norm and Post-Norm by introducing two coupled streams per layer. In this subsection, these two coupled streams are denoted as $\bar{X}_i$ and $\bar{Y}_i$, which is initialized by the input embeddings $\bar{X}_0 = \bar{Y}_0 = X$. For the ℓ-th block, SiameseNorm conduct the following update:

$$\tilde{Y}_\ell = \text{RMSNorm}(\bar{Y}_\ell), \quad O_\ell = \text{UniMixer}(\bar{X}_\ell + \tilde{Y}_\ell)$$
$$\bar{X}_{\ell+1} = \text{RMSNorm}(\bar{X}_\ell + O_\ell), \quad \bar{Y}_{\ell+1} = \bar{Y}_\ell + O_\ell.$$

For the M-th UniMixer block, $\bar{X}_\ell$ and $\bar{Y}_\ell$ are fused to generate the final representation, which is formulated as

$$X_{\text{output}} = \bar{X}_M + \text{RMSNorm}(\bar{Y}_M). \tag{20}$$

4.5 UniMixer Training Strategies

To require sparsity in the parameter matrices W_G and W^i_B, we introduce a temperature coefficient to control their sparsity level. However, a smaller temperature leads to sparser weights, while also resulting in the gradients to become sparse, weak, or even unstable. This can make the training process difficult, and optimization get trapped in local optima. On the other hand, our experiments show that the sparsity of the weight parameters has a significantly positive effect on model performance, as shown in Table 3 (subsection 5.3). Therefore, such sparsity is indispensable. A commonly used approach is to apply linear temperature annealing during the training process: starting with a relatively high initial temperature (e.g., $\tau = 1.0$) and gradually annealing it linearly to 0.05 as the number of training iterations increases, which is formulated as

$$\tau_j = \max\left\{\tau_{\text{start}} - \frac{(\tau_{\text{start}} - \tau_{\text{end}})j}{J}, \tau_{\text{end}}\right\}, \tag{21}$$

where τ_j is the j-th temperature coefficient, τ_{start} and τ_{end} are the initial temperature and final temperature, respectively, J is the iteration range for temperature annealing. When the amount of data is insufficient, linear annealing may lead to inadequate exploration in the early stage with a high temperature coefficient, or suboptimal optimization in the later stage with a low temperature coefficient. To address this, we can first use a high temperature coefficient (e.g., $\tau = 1.0$) to cold-start the training of the model; once the model is well trained, we then lower the temperature coefficient (e.g., $\tau = 0.05$) and retrain the low-temperature model using the weights of the high-temperature model as initialization.

5 Experiments

In this section, we conduct extensive experiments to compare the performance of the present UniMixer architecture with existing state-of-the-art (SOTA) approaches and to answer the following questions:

- **Q1**: Does the scaling efficiency of the UniMixer architecture outperform the SOTA architecture?
- **Q2**: How does the performance of the proposed method change under different settings of global and local mixing pattern?
- **Q3**: Does the lightweigh module, UniMixing-Lite, further improve the scaling efficiency?
- **Q4**: When deployed in a real-world online system, does UniMixer/UniMixing-Lite improve business metrics in A/B testing?

5.1 Experimental Setup

Datasets and Evaluation Metrics. We use the logged data from the real-world training dataset of the advertising delivery scenario on Kuaishou to model user retention and conduct the offline and online evaluation. The dataset contains over 0.7 billion user samples collected over one year, which comprises hundreds of heterogeneous features such as numerical features, ID features, cross features, and sequential features. A binary label (User Retention = 1/0) indicates whether the user returns to the Kuaishou application on the day following the users' first activation. For the scaling evaluation metrics of recommendation models, we adopt the two common metrics used in recommender system, i.e., area under the ROC curve (AUC), and UAUC (User-Level AUC), to evaluate the model performance, and dense parameter count, FLOPs, and MFU to evaluate the model efficiency.

Baselines and Experimental Details. We compare the present 2-blocks/4-blocks UniMixer/UniMixing-Lite architectures with the following representative SOTA frameworks, categorized by modeling paradigm

- **Attention-Based Architectures**: Heterogeneous Attention [1], HiFormer [1], and FAT [2], which use the field-specific query, key, and value projections to achieve heterogeneous feature interaction.
- **TokenMixer-Based Framework**: RankMixer [4] and TokenMixer-Large [5], which employ the rule-based token mixing operation to perform the feature interaction.
- **FM-Based Framework**: Wukong [6], which concatenates the outputs of a FMB and a linear projection layer to upscale the interaction component.

All experiments are conducted in a hybrid distributed training framework composed of 40 GPUs. All models use consistent optimizer hyperparameters: both the dense and sparse parts are optimized with Adam, with a learning rate set to 0.001.

5.2 Performance Comparison (for Q1)

The SOTA scaling architectures with approximately 100 million parameters are used to compare with UniMixer and UniMixer-Lite to explore their scaling laws. The heterogeneous attention architecture is used to be the base model. The main performance results of our models and the SOTA models are provided in Table 2. It can be observed that, under smaller parameter budgets and computational costs, both UniMixer and UniMixer-Lite architectures significantly outperform other SOTA models across multiple metrics.

Table 2: Performance and efficiency of $\sim$ 100M-parameter **UniMixer** and SOTA models in ad serving scenarios.

Model	**User Retention**				**Efficiency**	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
Heterogeneous Attention [1]	0.744577	–	0.733829	–	132.7M	1.68T
HiFormer [1]	0.741685	-0.2892%	0.731086	-0.2743%	107.5M	1.37T
Wukong [6]	0.744477	-0.0100%	0.733849	0.0020%	107.1M	1.40T
FAT [2]	0.744883	0.0306%	0.734280	0.0451%	138.4M	1.83T
RankMixer [4]	0.749329	0.4752%	0.738938	0.5109%	135.5M	1.68T
TokenMixer-Large [5]	0.748410	0.3833%	0.737940	0.4111%	103.3M	1.27T
UniMixer-2-Blocks 67.5M (Ours)	0.749770	0.5193%	0.739331	0.5502%	67.5M	2.07T
UniMixer-2-Blocks 101.5M (Ours)	0.750238	0.5661%	0.739983	0.6154%	101.5M	2.50T
UniMixer-Lite-2-Blocks 42.4M (Ours)	0.751121	0.6544%	0.740739	0.6910%	42.4M	2.17T
UniMixer-Lite-2-Blocks 76.2M (Ours)	0.751401	0.6824%	0.741215	0.7386%	76.2M	2.60T
UniMixer-Lite-4-Blocks 38.2M (Ours)	**0.752327**	**0.7750%**	**0.742091**	**0.8190%**	38.2M	1.26T
UniMixer-Lite-4-Blocks 84.5M (Ours)	**0.752718**	**0.8141%**	**0.742530**	**0.8701%**	84.5M	4.24T

Subsequently, in this advertising delivery scenario, the performance of RankMixer outperforms all other SOTA models except UniMixer/UniMixer-Lite. Therefore, we select the strongest SOTA model together with UniMixer/UniMixer-Lite for a scaling laws comparison. All models are trained on the same dataset with consistent hyperparameters. Their scaling curves with respect to parameters and

FLOPs are given in Figure 4. As observed, the AUC of all three models exhibits clear power-law trends as the number of parameters/FLOPs increases. UniMixer-Lite achieve the best scaling efficiency and exhibits a steeper improvement slope. According to the relationship between parameter count and AUC as shown in Fig. 4, the well-behaved scaling laws between AUC and Parameters/FLOPs for RankMixer, UniMixer and UniMixer-Lite can be formulated as follows

$$\Delta\text{AUC}_{\text{RankMixer}} = 0.002718\text{Params}^{0.116043}, \ \Delta\text{AUC}_{\text{RankMixer}} = 0.002022\text{FLOPs}^{0.116635},$$
$$\Delta\text{AUC}_{\text{UniMixer}} = 0.003032\text{Params}^{0.131973}, \ \Delta\text{AUC}_{\text{UniMixer}} = 0.002058\text{FLOPs}^{0.125702},$$
$$\Delta\text{AUC}_{\text{UniMixer-Lite}} = 0.003767\text{Params}^{0.141903}, \ \Delta\text{AUC}_{\text{UniMixer-Lite}} = 0.002338\text{FLOPs}^{0.135327}.$$

Among two constants in the scaling laws, the scaling exponent constant has the most significant impact on performance growth, which is the dominant factor in scaling efficiency. UniMixer-Lite demonstrates the strongest scaling efficiency, achieving both the largest scaling exponent and coefficient across parameters and FLOPs. This indicates that it benefits the most from increased model capacity.

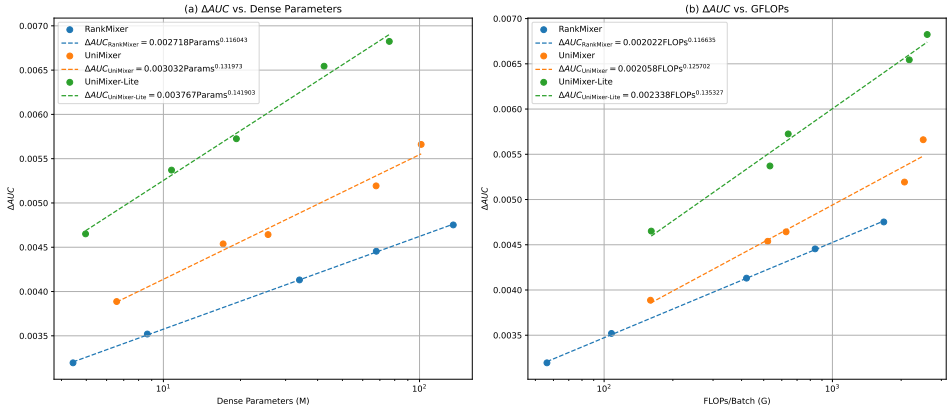

Figure 4: The scaling laws between AUC and Parameters/FLOPs for UniMixer-2-Blocks/UniMixer-Lite-2-Blocks and RankMixer architectures. The x-axis is presented on a logarithmic scale.

5.3 Ablation Studies (for Q2)

To explore the properties of global and local mixing weights, as well as the contribution of each module in UniMixer to AUC gains, we conduct ablation studies on various UniMixer variants and measure their relative AUC changes compared to the full UniMixer model. All variants are trained under similar settings. The results are shown in Table 3, which illustrate that removing any module or violating any parameter constraints leads to performance degradation, with low temperature coefficient and model warm-up having the most significant impact on overall performance.

Table 3: Ablation on components of UniMixer 6.57M.

Setting	User Retention				Efficiency	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
UniMixer	0.748464	–	0.738017	–	6.57M	0.16T
w/o Temperature Coefficient	0.746819	-0.1645%	0.736527	-0.1490%	6.57M	0.16T
w/o Symmetry Constraint	0.747891	-0.0573%	0.737447	-0.0570%	6.57M	0.16T
w/o Block-Specific Local Mixing Weight	0.748028	-0.0436%	0.737770	-0.0240%	6.53M	0.16T
w/o Model Warm-Up	0.747608	-0.0856%	0.737180	-0.0837%	6.57M	0.16T
SiameseNorm → Post Norm	0.748191	-0.0273%	0.737660	-0.0357%	6.56M	0.16T

5.4 Performance of the UniMixing-Lite Module (for Q3)

According to the scaling trends from Fig. 4, it can be obaserved that the present UniMixing-Lite architecture possesses the best parameter efficiency and computational efficiency. Here, we conduct

experiments to investigate the effects of different basis numbers b for $\{Z_\ell\}_{\ell=1}^{b}$, different rank r for A_G and B_G and different UniMixer block number. As shown in Table 4, as the basis numbers b and the rank r for A_G, B_G increase, the model performance improves accordingly. However, in terms of parameter efficiency, increasing the number of basis b yields a higher AUC gain than increasing the rank r. To observe the effects of the low-rank approximation $A_G B_G$ and basis matrices $\{Z_\ell\}_{\ell=1}^{b}$ with the Sinkhorn–Knopp operation on reconstructing the global and local mixing matrices, in a 2-blocks-UniMixer-Lite architecture, we visualize the reconstructed global matrix $\bar{W}_G$ and the first six local mixing matrices $\bar{W}_B^i$ of the first UniMixer block with the temperature coefficients $\tau = 1$ and $\tau = 0.05$, as shown in Fig. 5. The input embedding dimension is 768, and the block size is 6; therefore, we have $\bar{W}_G \in \mathbb{R}^{128\times128}$ and $\bar{W}_B^i \in \mathbb{R}^{6\times6}$, where $A_G \in \mathbb{R}^{128\times16}$ and $B_G \in \mathbb{R}^{16\times128}$. According to Fig. 5, although the low-rank approximation and basis matrices are used in the module, the Sinkhorn–Knopp operation can still ensure that the matrix remains close to full rank. In addition, compared with Figs. 5(a)(b) and (c)(d), the global and local mixing matrices with a lower temperature coefficient exhibit sharper interaction distributions than those with a higher temperature coefficient. From the ablation results given in the ablation studies, we can conclude that the sparsity of $\bar{W}_G$ and $\bar{W}_B^i$ leads to a significant improvement in model performance.

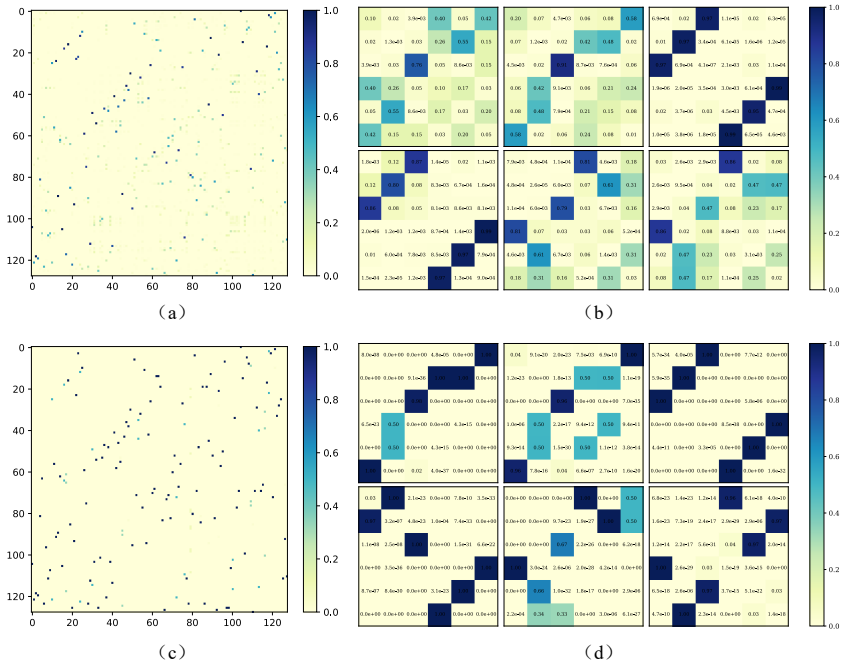

Figure 5: $\bar{W}_G$ and $\bar{W}_B^i$ of UniMixer-Lite with different temperature coefficients. (a) $\bar{W}_G$ with $\tau = 1$; (b) $\{\bar{W}_B^i\}_{i=1}^{6}$ with $\tau = 1$; (c) $\bar{W}_G$ with $\tau = 0.05$; (d) $\{\bar{W}_B^i\}_{i=1}^{6}$ with $\tau = 0.05$.

On the other hand, according to Table 4, it can be observed that as the depth of UniMixer increases, the developed model continues to exhibit a clear scaling-up trend, whereas RankMixer shows a performance degradation as the RankMixer blocks are stacked. The scaling curves of UniMixing-Lite with 2 blocks and 4 blocks are shown in Fig. 6, which implies that scaling along depth is more efficient than scaling along width.

5.5 Online A/B Test Results (for Q4)

To verify the online performance of the proposed UniMixer architecture, we have deployed UniMixer and UniMixer-Lite across multiple advertising delivery scenarios on Kuaishou. In the online A/B test, we measure user engagement using the Cumulative Active Days (CAD) over a 30-day observation window, excluding the installation day (day 0). Across multiple scenarios, CAD of D1-D30 increased by more than 15% on average.

Table 4: Effects of the basis number, rank, and UniMixer block number in UniMixing-Lite.

Basis Number	**User Retention**				**Efficiency**	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
b=2	0.749228	–	0.738776	–	4.968M	0.161T
b=4	0.750230	0.1002%	0.739876	0.1100%	4.973M	0.161T
b=8	0.750283	0.1055%	0.739854	0.1078%	4.98M	0.161T
Rank	**User Retention**				**Efficiency**	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
r=2	0.748568	–	0.738177	–	4.4525M	0.160T
r=64	0.749002	0.0434%	0.738604	0.0427%	4.705M	0.160T
r=128	0.749228	0.0660%	0.738776	0.0599%	4.9675M	0.161T
r=256	0.749539	0.0971%	0.739437	0.1260%	5.4925M	0.163T
Block Number	**User Retention**				**Efficiency**	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
RankMixer-2-Blocks	0.747772	–	0.737322	–	4.4375M	0.056T
RankMixer-4-Blocks	0.746706	-0.1066%	0.736018	-0.1304%	8.6575M	0.108T
UniMixer-Lite-2-Blocks	0.749228	–	0.738776	–	4.9675M	0.161T
UniMixer-Lite-4-Blocks	0.750803	0.1575%	0.740594	0.1818%	9.715M	0.316T
UniMixer-Lite-8-Blocks	0.750875	0.1647%	0.740602	0.1826%	19.2125M	0.629T

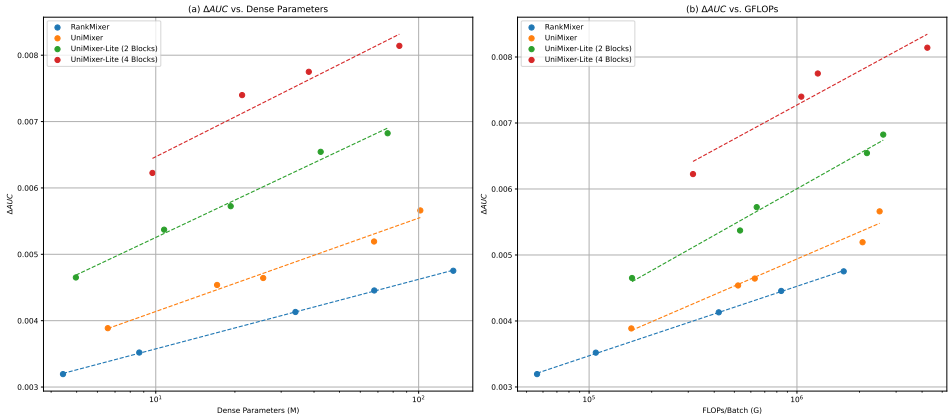

Figure 6: The scaling curves between AUC and Parameters/FLOPs for RankMixer and UniMixer/UniMixer-Lite with 2 blocks and 4 blocks.

6 Conclusions

In this work, a unified scaling framework is established for scaling laws in recommendation systems, which ridges the connections among attention-based, TokenMixer-based, and FM-based methods and makes it possible to leverage their respective strengths. From the obtained scaling laws, compared with the SOTA architectures, the present UniMixer-Lite achieved the best parameter efficiency and computational efficiency. We have deployed the architectures across multiple scenarios at Kuaishou, yielding significant offline and online gains. This work no longer treats existing scaling blocks (e.g., Heterogeneous Attention, TokenMixer, Wukong) in recommender in isolation. Instead, it establishes a unified theoretical framework that provides guidance for scaling design in recommendation systems. We believe that the unified architecture can help the recommendation systems community achieve its own "attention moment". The unified module, UniMixer, serves as a fundamental block tailored for the recommendation domain, whose applicability can be further extended to user behavior sequence modeling and generative recommendation tasks.

References

[1] Huan Gui, Ruoxi Wang, Ke Yin, Long Jin, Maciej Kula, Taibai Xu, Lichan Hong, and Ed H Chi. Hiformer: Heterogeneous feature interactions learning with transformers for recommender systems. *arXiv preprint arXiv:2311.05884*, 2023.

[2] Bencheng Yan, Yuejie Lei, Zhiyuan Zeng, Di Wang, Kaiyi Lin, Pengjie Wang, Jian Xu, and Bo Zheng. From scaling to structured expressivity: Rethinking transformers for ctr prediction. *arXiv preprint arXiv:2511.12081*, 2025.

[3] Liren Yu, Wenming Zhang, Silu Zhou, Tao Zhang, Zhixuan Zhang, and Dan Ou. Hhft: Hierarchical heterogeneous feature transformer for recommendation systems. *arXiv preprint arXiv:2511.20235*, 2025.

[4] Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, et al. Rankmixer: Scaling up ranking models in industrial recommenders. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 6309–6316, 2025.

[5] Yuchen Jiang, Jie Zhu, Xintian Han, Hui Lu, Kunmin Bai, Mingyu Yang, Shikang Wu, Ruihao Zhang, Wenlin Zhao, Shipeng Bai, et al. Tokenmixer-large: Scaling up large ranking models in industrial recommenders. *arXiv preprint arXiv:2602.06563*, 2026.

[6] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. Wukong: Towards a scaling law for large-scale recommendation. *Proceedings of the 41st International Conference on Machine Learning*, 235:1–20, 2024.

[7] Bojian Hou, Xiaolong Liu, Xiaoyi Liu, Jiaqi Xu, Yasmine Badr, Mengyue Hang, Sudhanshu Chanpuriya, Junqing Zhou, Yuhang Yang, Han Xu, et al. Kunlun: Establishing scaling laws for massive-scale recommendation systems through unified architecture design. *arXiv preprint arXiv:2602.10016*, 2026.

[8] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

[9] Qin Ding, Kevin Course, Linjian Ma, Jianhui Sun, Rouchen Liu, Zhao Zhu, Chunxing Yin, Wei Li, Dai Li, Yu Shi, et al. Bending the scaling law curve in large-scale recommendation systems. *arXiv preprint arXiv:2602.16986*, 2026.

[10] Xiao Lv, Jiangxia Cao, Shijie Guan, Xiaoyou Zhou, Zhiguang Qi, Yaqiang Zang, Ben Wang, and Guorui Zhou. Marm: Unlocking the recommendation cache scaling-law through memory augmentation and scalable complexity. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 2022–2031, New York, NY, USA, 2025. Association for Computing Machinery.

[11] Zhaoqi Zhang, Haolei Pei, Jun Guo, Tianyu Wang, Yufei Feng, Hui Sun, Shaowei Liu, and Aixin Sun. Onetrans: Unified feature interaction and sequence modeling with one transformer in industrial recommender. *arXiv preprint arXiv:2510.26104*, 2025.

[12] Songpei Xu, Shijia Wang, Da Guo, Xianwen Guo, Qiang Xiao, Bin Huang, Guanlin Wu, and Chuanjiang Luo. Climber: Toward efficient scaling laws for large recommendation models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, page 6193–6200. Association for Computing Machinery, 2025.

[13] Yunwen Huang, Shiyong Hong, Xijun Xiao, Jinqiu Jin, Xuanyuan Luo, Zhe Wang, Zheng Chai, Shikang Wu, Yuchao Zheng, and Jingjian Lin. Hyformer: Revisiting the roles of sequence modeling and feature interaction in ctr prediction. *arXiv preprint arXiv:2601.12681*, 2026.

[14] Lee Xiong, Zhirong Chen, Rahul Mayuranath, Shangran Qiu, Arda Ozdemir, Lu Li, Yang Hu, Dave Li, Jingtao Ren, Howard Cheng, et al. Llatte: Scaling laws for multi-stage sequence modeling in large-scale ads recommendation. *arXiv preprint arXiv:2601.20083*, 2026.

[15] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[16] Xintian Han, Honggang Chen, Quan Lin, Jingyue Gao, Xiangyuan Ren, Lifei Zhu, Zhisheng Ye, Shikang Wu, XiongHang Xie, Xiaochu Gan, et al. Lemur: Large scale end-to-end multimodal recommendation. *arXiv preprint arXiv:2511.10962*, 2025.

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

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

[19] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1725–1731, 2017.

[20] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1161–1170, 2019.

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

[22] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Ed H Chi, and Min Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale ctr prediction. In *Proceedings of the Web Conference 2021*, pages 1785–1797, 2021.

[23] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[24] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2022.

[25] Xianyang Qi, Yuan Tian, Zhaoyu Hu, Zhirui Kuai, Chang Liu, Hongxiang Lin, and Lei Wang. Mtmixatt: Integrating mixture-of-experts with multi-mix attention for large-scale recommendation, 2025.

[26] Tianyu Li, Dongchen Han, Zixuan Cao, Haofeng Huang, Mengyu Zhou, Ming Chen, Erchao Zhao, Xiaoxi Jiang, Guanjun Jiang, and Gao Huang. Siamesenorm: Breaking the barrier to reconciling pre/post-norm, 2026.

A A numerical example of equivalent transformation of TokenMixer

The following input hidden state $X \in \mathbb{R}^{2\times 6}$ is given

$$X = \left[\begin{array}{ccc:ccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{array}\right],$$

where x_i is a scalar. Then the input hidden state X passed through the TokenMixer operation is transformed into

$$\text{TokenMixer}(X) = \left[\begin{array}{ccc} \frac{x_1}{x_4} & \frac{x_2}{x_5} & \frac{x_3}{x_6} \\ \hdashline \frac{x_7}{x_{10}} & \frac{x_8}{x_{11}} & \frac{x_9}{x_{12}} \end{array}\right].$$

The output of TokenMixer can be flatten as a vector

$$\text{flatten}(\text{TokenMixer}(X)) = [x_1, x_2, x_3, x_7, x_8, x_9, x_4, x_5, x_6, x_{10}, x_{11}, x_{12}]^\mathsf{T} \tag{22}$$

On the other hand, the vector $\text{flatten}(X)$ can be transformed into $\text{flatten}(\text{TokenMixer}(X))$ by multiplying a 12×12 matrix, which can be formulated as

$$\underbrace{\left[\begin{array}{ccc:ccc:ccc:ccc} 1&0&0&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0&0&0 \\ \hdashline 0&0&0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&0&1&0&0&0 \\ \hdashline 0&0&0&1&0&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0&0&0 \\ \hdashline 0&0&0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{array}\right]}_{W^{\text{perm}}} \underbrace{\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{array}\right]}_{\text{fllaten}(X)} = \underbrace{\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_7 \\ x_8 \\ x_9 \\ x_4 \\ x_5 \\ x_6 \\ x_{10} \\ x_{11} \\ x_{12} \end{array}\right]}_{\text{flatten}(\text{TokenMixer}(X))} \tag{23}$$

According to (22) and (23), the TokenMixer operation in this numerical example is equivalently transformed into the form of multiply matrix. In addition, the permutation Matrix $W^{\text{perm}} \in \mathbb{R}^{12\times 12}$ can be equivalently decomposed into the Kronecker product of the follwing two small matrices

$$W^{\text{perm}} = \underbrace{\left[\begin{array}{cccc} 1&0&0&0 \\ 0&0&1&0 \\ 0&1&0&0 \\ 0&0&0&1 \end{array}\right]}_{\text{Global Mixing Matrix}} \otimes \underbrace{\left[\begin{array}{ccc} 1&0&0 \\ 0&1&0 \\ 0&0&1 \end{array}\right]}_{\text{Local Mixing matric}}.$$

B The computation pipeline optimization of the UniMixing module

Define $W_G \in \mathbb{R}^{(L//B)\times L//B}$ and $W_B^i \in \mathbb{R}$ as follows

$$W_G = \left[\begin{array}{ccc} w^G_{(1,1)} & \cdots & w^G_{(1,L//B)} \\ \cdots & \cdots & \cdots \\ w^G_{(L//B,1)} & \cdots & w^G_{(L//B,L//B)} \end{array}\right], W_B^i = \left[\begin{array}{ccc} v^i_{(1,1)} & \cdots & v^i_{(1,B)} \\ \cdots & \cdots & \cdots \\ v^i_{(B,1)} & \cdots & v^i_{(B,B)} \end{array}\right], \tag{24}$$

where w_{ij} and v_{ij} are the scalars. According to (12), $\text{flatten}(X)$ is evenly split into $L//B$ vectors, which is rewritten as

$$\text{flatten}(X) = \left[\boldsymbol{x}_1 |\ \boldsymbol{x}_2 \mid \ \ldots \ \mid \boldsymbol{x}_{\frac{L}{B}}\right]^\mathsf{T}, \tag{25}$$

where $\boldsymbol{x}_i$ is a row vector of dimension B.

According to the origin expression of UniMing (11), the term $\Big(W_G \otimes \{W_B^i\}_{i=1}^{L//B}\Big)\text{flatten}(X)$ can be rewritten as

$$\begin{aligned}\Big(W_G \otimes \{W_B^i\}_{i=1}^{L//B}\Big)\text{flatten}(X) &= \begin{bmatrix} w^G_{(1,1)}W_B^1 & \dots & w^G_{(1,L//B)}W_B^{L//B} \\ \dots & \dots & \dots \\ w^G_{(L//B,1)}W_B^1 & \dots & w^G_{(L//B,L//B)}W_B^{L//B} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}}^\mathsf{T} \end{bmatrix} \\ &= \begin{bmatrix} w^G_{(1,1)}W_B^1\boldsymbol{x}_1^\mathsf{T} + \ldots + w^G_{(1,L//B)}W_B^{L//B}\boldsymbol{x}_{\frac{L}{B}}^\mathsf{T} \\ \dots \\ w^G_{(L//B,1)}W_B^1\boldsymbol{x}_1^\mathsf{T} + \ldots + w^G_{(L//B,L//B)}W_B^{L//B}\boldsymbol{x}_{\frac{L}{B}}^\mathsf{T} \end{bmatrix} \in \mathbb{R}^{L\times 1}\end{aligned} \tag{26}$$

On the other hand, we can obtain the following expression

$$\begin{aligned} & W_G\text{reshape}\Big(\Big[\boldsymbol{x}_1 W_B^{1\mathsf{T}} \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}}\Big], \frac{L}{B}, B\Big) \\ &= \begin{bmatrix} w^G_{(1,1)} & \dots & w^G_{(1,L//B)} \\ \dots & \dots & \dots \\ w^G_{(L//B,1)} & \dots & w^G_{(L//B,L//B)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 W_B^{1\mathsf{T}} \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}} \end{bmatrix} \\ &= \begin{bmatrix} w^G_{(1,1)}\boldsymbol{x}_1 W_B^{1\mathsf{T}} + \ldots + w^G_{(1,L//B)}\boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}} \\ \dots \\ w^G_{(L//B,1)}\boldsymbol{x}_1 W_B^{1\mathsf{T}} + \ldots + w^G_{(L//B,L//B)}\boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}} \end{bmatrix} \in \mathbb{R}^{\frac{L}{B}\times B} \end{aligned} \tag{27}$$

The element in ((26)) and the element in (27) satisfy

$$w^G_{(i,1)}W_B^1\boldsymbol{x}_1^\mathsf{T} + \ldots + w^G_{(i,L//B)}W_B^{L//B}\boldsymbol{x}_{\frac{L}{B}}^\mathsf{T} = (w^G_{(i,1)}\boldsymbol{x}_1 W_B^{1\mathsf{T}} + \ldots + w^G_{(i,L//B)}\boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}})^\mathsf{T}, \tag{28}$$

which results in

$$\Big(W_G\otimes\{W_B^i\}_{i=1}^{L//B}\Big)\text{flatten}(X) = \text{reshape}\Big(W_G\text{reshape}\Big(\Big[\boldsymbol{x}_1 W_B^{1\mathsf{T}} \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}\mathsf{T}}\Big], \frac{L}{B}, B\Big), L, 1\Big)$$

Since both W_B^i and $W_B^{i\mathsf{T}}$ are learnable parameters, the transpose of the parameter does not affect the model. Therefore, the UniMixing module after the computation pipeline optimization can be formulated as

$$\text{UniMixing}(X) = \text{reshape}\Big(W_G\text{reshape}\Big(\Big[\boldsymbol{x}_1 W_B^1 \;\Big|\; \boldsymbol{x}_2 W_B^2 \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W_B^{\frac{L}{B}}\Big], \frac{L}{B}, B\Big), 1, L\Big). \tag{29}$$