nUNIMIXER: BREAKING THE BARRIER TO A UNIFIED AND SCALABLE BACKBONE FOR RECOMMENDER SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

In recent years, the scaling laws of recommendation models have attracted increasing attention. Three mainstream architectures have emerged for scaling ranking models: attention-based, TokenMixer-based, and factorization-machine-based architectures. These architectures differ fundamentally in both their design philosophies and structural characteristics. In this paper, we propose **UniMixer**, a unified theoretical framework for model scaling that establishes connections among these three mainstream blocks. By parameterizing the rule-based TokenMixer, we develop a Dual-Axis Heterogeneous Mixer (**DAHM**) that learns token-mixing patterns during training and achieves global element-wise feature mixing. To relax the constraints on the parameter matrices and improve the scaling efficiency, we further develop a normalized UniMixer block, termed **nUniMixer**. Extensive offline and online experiments across diverse tasks and application scenarios demonstrate the superior scalability of **nUniMixer**.

1 INTRODUCTION

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

Based on a large number of multi-field user and item features, recommender systems predict the user behavior to present the most relevant content to them to increase users' 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 feature 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 to 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 Gui et al. (2023), FAT Yan et al. (2025), and HHFT Yu et al. (2025), 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 Zhu et al. (2025), TokenMixer-Large Jiang et al. (2026), etc.) employ the rule-based token mixing operation to achieve heterogeneous feature interactions. Factorization-machine-based methods (e.g. Wukong Zhang et al. (2024), Kunlun Hou et al. (2026), etc.), on the other hand, model feature interactions by introducing a Factorization Machine (FM) block to compute interactions among the input embeddings within

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

each layer. These frameworks are built upon completely different scaling blocks, yet all demonstrate the capability to scale up model performance. More related work is summarized in Appendix A. 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 gaps 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 and develop a more efficient scaling framework. Our main contributions can be summarized as follows: (1) We reveal the feature interaction patterns of TokenMixer via equivalent parameterization. The properties of the parameter matrix, such as compressibility, doubly stochasticity, sparsity, and symmetry, are revealed from a theoretical perspective. (2) We propose a unified scaling framework, termed UniMixer, which bridges the 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. Meanwhile, the present Dual-Axis Heterogeneous Mixer (DAHM) can achieve global element-wise feature mixing. (3) To address the problem of complicated training strategies caused by strong constraints of parameter matrices, we develop a normalized UniMixer (nUniMixer) module, which constrains each column of the parameter matrices to the closed unit ball and normalizes hidden states onto the unit hypersphere, improving training stability and scaling efficiency. (4) Extensive offline and online experiments are performed to demonstrate the superior scalability of UniMixer and nUniMixer.

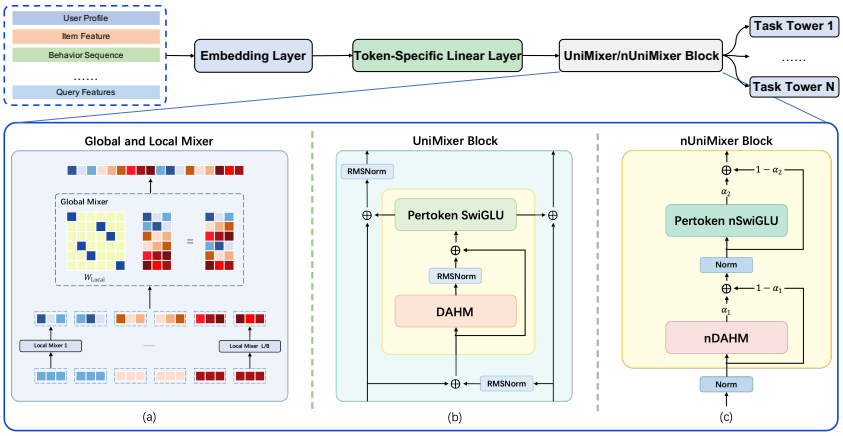


Figure 1: The UniMixer/nUniMixer architectures for scaling laws in recommendation systems. (a) The local and global mixer module; (b) The UniMixer block; (c) The nUniMixer block.

2 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 $\mathbb{D} = \{(\mathbf{X}_1, y_1), \ldots, (\mathbf{X}_i, y_i), \ldots, (\mathbf{X}_S, y_S)\}$, 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, S is the number of data points. In general, the input features $\mathbf{X} = \{\mathbf{X}^{\mathcal{C}}, \mathbf{X}^{\mathcal{D}}\}$ are divided into categorical features $\mathbf{X}^{\mathcal{C}}$ and dense features $\mathbf{X}^{\mathcal{D}}$. $|\mathcal{C}|$ and $|\mathcal{D}|$ 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 (Gui et al., 2023). To date, scaling laws in the recommendation domain have primarily been established through three types of foundational blocks and their variants: attention-based (Gui et al., 2023; Yan et al., 2025; Yu et al., 2025), TokenMixer-based (Zhu et al., 2025; Jiang et al., 2026; Qi et al., 2025), and FM-based (Zhang et al., 2024; Hou et al., 2026) architectures. The details of these three foundational blocks are provided in Appendix B. In this work, we focus on establishing a unified theoretical framework for recommendation systems

that bridges the connections current mainstream scaling blocks. Based on the unified theoretical framework, the advanced scaling architecture is designed to further increase the scaling ROI. As shown in Fig. 1, the overall architecture consists of feature tokenization, M UniMixer/nUniMixer blocks, and N prediction task towers.

3 UNIMIXER

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

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

Each feature domain is mapped to an embedding vector with dimension d_{domain} through an embedding layer $\mathbf{e}_n = \text{Embedding}(\mathbf{X}_{\text{domain}}) \in \mathbb{R}^{d_{\text{domain}}}$, where $\mathbf{X}_{\text{domain}}$ denotes all the features within a feature domain, and d_{domain} is the embedding dimension corresponding to this feature domain. The obtained feature domain embeddings are concatenated into one embedding vector $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{N_f}]$. Similar to the approach of Zhu et al. (2025), the embedding vector $\mathbf{E}$ is evenly divided into an appropriate number of blocks. Then, each block is projected into a token embedding using the token-specific linear layer as $\boldsymbol{x}_i = W_i^{\text{proj}} \mathbf{E}_{di:di+d} + \mathbf{b}_i^{\text{proj}} \in \mathbb{R}^D$, 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 $\boldsymbol{x}_i$ row-wise, where T is the number of tokens.

3.1 UNIMIXER BLOCK

Heterogeneous Feature Interactions. 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. 2(a).

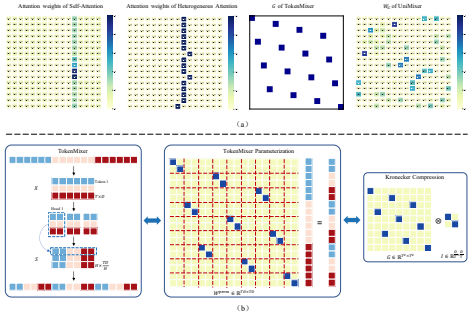


Figure 2: (a) The global mixing weights of different methods. (b) Equivalent parameterization of the rule-based TokenMixer.

As illustrated in the Fig. 2(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. 2(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 weights 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 the number of tokens to equal the number of heads further restricts the selection of heterogeneous feature interaction patterns. Through in-depth analysis of the TokenMixer operation, we have made the following interesting findings (Theorems 3.1 and 3.2), which make it possible to parameterize the TokenMixer operation.

Theorem 3.1 (Equivalent parameterization of TokenMixer)**.** *Let $X \in \mathbb{R}^{T \times D}$ be the input hidden state. If the hidden state is processed by the TokenMixer operation, then there exists a permutation matrix W^{perm} such that TokenMixer(X) = reshape$(W^{perm}$vec$(X))$, where $W^{perm} \in \mathbb{R}^{TD \times TD}$ is a large permutation matrix.*

The proof of Theorem 3.1 is given in Appendix D. A concrete numerical example is provided in Appendix C. Based on Theorem 3.1, 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 and analysis, we have some findings regarding the permutation matrix W^{perm} of TokenMixer, which can be formally described by Theorem 3.2. The properties of W^{perm} can be summarized as compressibility, doubly stochasticity, sparsity, and symmetry. The corresponding proofs are given in Appendix D.

Theorem 3.2 (Properties of the permutation matrix W^{perm})**.** *Given the input hidden state X, the number of tokens T, and the dimension of the token D, then the permutation matrix W^{perm} satisfies the following properties:*

- ***Compressibility.*** *The permutation matrix W^{perm} can be equivalently decomposed into the Kronecker product of two smaller matrices, i.e., $W^{perm} = G \otimes I$, 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., $\sum_{p=1}^{TD} w_{pq}^{perm} = 1, \sum_{q=1}^{TD} w_{pq}^{perm} = 1$.*
- ***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., $W^{perm} = W^{perm\mathsf{T}}$. If $T \neq H$, the permutation matrix W^{perm} is an asymmetric matrix.*

According to the compressibility of $W^{\text{perm}} = G \otimes I$, 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 for establishing a unified scaling module: (1) Directly using parameterized G and I to reconstruct W^{perm} still produces an intermediate variable of size $[TD, TD]$ during model training and inference, 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.

The Unified Token Mixing Module. 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_{Global}. Considering the sparsity of the permutation matrix and to achieve sufficient heterogeneous feature interactions, we assign a distinct parameterized weight W_{Local}^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_{Global} and W_{Local}^i, which is formulated as

$$\text{UniMixing}(X) = \text{reshape}\Big(\Big(W_{\text{Global}} \otimes \{W_{\text{Local}}^i\}_{i=1}^{L/B}\Big)\text{vec}(X), 1, L\Big), \tag{2}$$

where $\otimes$ is the generalized Kronecker product. Next, the computation pipeline of (2) is optimized to significantly reduce both the computational cost and GPU memory requirements. The embedding vector $\text{vec}(X)$ is first evenly split into L/B vectors and the size of each vector is B, which is expressed as

$$\Big[\boldsymbol{x}_1 | \, \boldsymbol{x}_2 \mid \ldots \mid \boldsymbol{x}_{\frac{L}{B}}\Big] = \text{Split}\Big(\text{vec}(X), \frac{L}{B}\Big). \tag{3}$$

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

$$H_s = \text{reshape}\Big(\Big[\boldsymbol{x}_1 W_{\text{Local}}^1 \,\Big|\, \boldsymbol{x}_2 W_{\text{Local}}^2 \,\Big|\, \ldots \,\Big|\, \boldsymbol{x}_{\frac{L}{B}} W_{\text{Local}}^{\frac{L}{B}}\Big], \frac{L}{B}, B\Big) = \begin{bmatrix} \boldsymbol{x}_1 W_{\text{Local}}^1 \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W_{\text{Local}}^{\frac{L}{B}} \end{bmatrix}. \tag{4}$$

Therefore, the output of UniMixing module can be obtained as

$$\text{UniMixing}(X) = \text{reshape}\Big(W_{\text{Global}} H_s, 1, L\Big). \tag{5}$$

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 (2) is provided in Appendix E. According to (4) and (5), W^i_{Local} controls the intra-block interaction pattern, while W_{Global} controls the inter-block interaction pattern. For the embedding inputs with dimension L, it is no longer required that the number of tokens is equal to the number of heads. 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 make all elements of W_{Global} and W^i_{Local} 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, $(W_{\text{Global}} + W^{\mathsf{T}}_{\text{Global}})/2$ and $(W^i_{\text{Local}} + W^{i\mathsf{T}}_{\text{Local}})/2$ are used to achieve the symmetry constraints of parameter matrices. The final constrained weights can be obtained by $\bar{W}_{\text{Global}} = \text{Sinkhorn-Knopp}\Big(\frac{W_{\text{Global}}+W^{\mathsf{T}}_{\text{Global}}}{2\tau}\Big)$, $\bar{W}^i_{\text{Local}} = \text{Sinkhorn-Knopp}\Big(\frac{W^i_{\text{Local}}+W^{i\mathsf{T}}_{\text{Local}}}{2\tau}\Big)$, where τ is the temperature coefficient.

A Unified Perspective of Heterogeneous Feature Interaction. Observing V_h in (8) and H in (4), we find that if the number of blocks L/B is set as T, and W^{ih}_V and W^i_{Local} have the same dimensions, then $H = V_h$. This implies that the local interaction projection of UniMixer is equivalent to the value projection of the heterogeneous attention layer under $W^i_V = W^i_{\text{Local}}$. On the other hand, the dimension and the role of W_{Global} are the same as the attention weights, except that W_{Global} needs to satisfy the doubly stochasticity, sparsity, and symmetry. The feature interaction of Wukong is based on the FM component. According to (10), the expression of $\text{FMB}(X)$ can be rewritten as $\text{FMB}(X) = \text{reshape}(\text{MLP}(\text{LN}(\text{vec}(XI(XI)^{\mathsf{T}}Y))))$, where I is the identity matrix with an appropriate dimension. Let us focus on the core feature interaction module $XI(XI)^{\mathsf{T}}Y$. 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

$$\text{UniMixing}(X) = \text{reshape}\Bigg(\underbrace{G(X, W_{\text{Global}})}_{\text{Global Mixing Pattern}} \underbrace{\begin{bmatrix} \boldsymbol{x}_1 W^1_{\text{Local}} \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}}_{\text{Local}} \end{bmatrix}}_{\text{Local Mixing Pattern}}, 1, L\Bigg), \tag{6}$$

where $G(X, W_{\text{Global}})$ 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 (6), their differences are summarized in Table 5 (see Appendix F). For the self-attention, heterogeneous attention, and FM, the global mixing pattern $G(X, W_{\text{Global}})$ is obtained by computing the inner-product similarity between two tokens. The global mixing pattern of TokenMixer is independent of the input token embedding.

Dual-Axis Heterogeneous Mixer. As given in (5), the UniMixing module first performs intra-block local mixing, followed by global coordination across blocks. Nevertheless, this process does not achieve element-wise mixing. We further develop a global element-wise feature mixing module, namely Dual-Axis Heterogeneous Mixer (DAHM). Considering the expression of the global and local mixer in UniMixing module (5), we can also assign a distinct global parameter matrix W^i_{Global} to each row of H^{T}_s by transposing $W_{\text{Global}} H_s$ into $(W_{\text{Global}} H_s)^{\mathsf{T}} = H^{\mathsf{T}}_s W^{\mathsf{T}}_{\text{Global}}$. Then, the DAHM block can be formulated as follows

$$\text{DAHM}(X; W^i_{\text{Local}}, W^j_{\text{Global}}) = \text{reshape}\Bigg(\begin{bmatrix} \bar{\boldsymbol{h}}_1 W^1_{\text{Global}} \\ \vdots \\ \bar{\boldsymbol{h}}_B W^B_{\text{Global}} \end{bmatrix}^{\mathsf{T}}, 1, L\Bigg), \quad \begin{bmatrix} \bar{\boldsymbol{h}}_1 \\ \vdots \\ \bar{\boldsymbol{h}}_B \end{bmatrix} = H^{\mathsf{T}}_s = \begin{bmatrix} \boldsymbol{x}_1 W^1_{\text{Local}} \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}}_{\text{Local}} \end{bmatrix}^{\mathsf{T}}.$$

With this operation, the DAHM module can achieve global element-wise mixing. A concrete numerical example is given in Appendix G. Inspired by Chen et al. (2026), the basis-composed

modules are introduced to dynamically generate the heterogeneous global and local mixing weight matrices $\{W^j_{\text{Global}}\}_{j=1}^{B}$ and $\{W^i_{\text{Local}}\}_{i=1}^{L/B}$. Define two sets of basis matrices for W^i_{Local} as $\{Z^\ell_{\text{Local}}\}_{\ell=1}^{b}$, local mixing weight vectors over these bases as $\{\boldsymbol{\omega}^i_{\text{Local}}\}_{i=1}^{L/B}$, $\{Z^\ell_{\text{Global}}\}_{\ell=1}^{b}$, global mixing weight vectors over these bases as $\{\boldsymbol{\omega}^i_{\text{Global}}\}_{i=1}^{L/B}$, where b is the number of the bases of local and global mixing weights and $\boldsymbol{\omega}^i_{\text{Local}} = [\omega^i_{\text{Local},1}, \ldots, \omega^i_{\text{Local},b}], \boldsymbol{\omega}^i_{\text{Global}} = [\omega^i_{\text{Global},1}, \ldots, \omega^i_{\text{Global},b}] \in \mathbb{R}^b$. The local and global mixing parameter matrices can be constructed as $W^i_{\text{Local}} = \sum_{\ell=1}^{b} \omega^i_{\text{Local},\ell} Z^\ell_{\text{Local}}, \quad W^i_{\text{Global}} = \sum_{\ell=1}^{b} \omega^i_{\text{Global},\ell} Z^\ell_{\text{Global}}$. To achieve the training stability and performance gains as model depth increases, SiameseNorm (Li et al., 2026b) is introduced into the UniMixer architecture. SiameseNorm resolves the tension between Pre-Norm and Post-Norm by introducing two coupled streams per layer. These two coupled streams are denoted as $\bar{X}_i$ and $\bar{Y}_i$, which are initialized by the input embeddings $\bar{X}_0 = \bar{Y}_0 = X$. For the ℓ-th block, SiameseNorm performs 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 $X_{\text{output}} = \bar{X}_M + \text{RMSNorm}(\bar{Y}_M)$.

4 nUniMixer

As mentioned in Section 3.1, the Sinkhorn-Knopp and the temperature annealing operations are used to ensure the doubly stochastic property of the mixer parameter matrices and sparsity, respectively, which leads to the complicated training strategy. To address this problem, the nUniMixer framework is developed to relax the constraints on $\{W^j_{\text{Global}}\}_{j=1}^{B}$ and $\{W^i_{\text{Local}}\}_{i=1}^{L/B}$. As shown in Fig. 1(c), the architecture consists of Normalized DAHM, Pertoken normalized SwiGLU, and the learnable residual update.

Normalized DAHM. Similar to (3), the embedding vector $\text{vec}(X)$ is first evenly split into L/B vectors and then stacked to construct a hidden state. Inspired by normalized Transformer (nGPT) (Loshchilov et al., 2025), this hidden state is normalized along its embedding dimension, which is formulated as $\bar{X} = \text{Norm}\Big(\text{reshape}\Big(\text{vec}(X), \frac{L}{B}, B\Big)\Big)$, where $\text{Norm}(\cdot)$ is the ℓ_2 norms along the block dimension. Differing from nGPT, we propose to normalize the global and local mixing parameter matrices $\{W^j_{\text{Global}}\}_{j=1}^{B}$ and $\{W^i_{\text{Local}}\}_{i=1}^{L/B}$ in DAHM, which is expressed as follows

$$\bar{W}^i_{\text{Local}} = \text{BallNorm}(W^i_{\text{Local}}), \quad \bar{W}^j_{\text{Global}} = \text{BallNorm}(W^j_{\text{Global}}),$$

where $\text{BallNorm}(\cdot)$ is applied column-wise and differs from conventional ℓ_2 normalization in that columns with ℓ_2 norms no greater than 1 remain unchanged. Therefore, the normalized DAHM (nDAHM) can be obtained as

$$\text{nDAHM}(X; \bar{W}^i_{\text{Local}}, \bar{W}^j_{\text{Global}}) = \text{BallNorm}\big(\text{DAHM}\big(\bar{X}; \bar{W}^i_{\text{Local}}, \bar{W}^j_{\text{Global}}\big)\big).$$

After nDAHM, the following learnable residual update is applied to the current model structure.

$$O = \text{Norm}\big((1-\alpha_1)\bar{X} + \alpha_1 \text{nDAHM}(X; \bar{W}^i_{\text{Local}}, \bar{W}^j_{\text{Global}})\big).$$

The residual Norm here and in the following SwiGLU block can optionally be replaced by scaling its argument with $\nu(\alpha) = \text{stopgrad}\left((1-2\alpha+2\alpha^2)^{-1/2}\right)$.

Pertoken normalized SwiGLU. Similar to Jiang et al. (2026), the pertoken SwiGLU is introduced to model the feature heterogeneity among different tokens. For the i-th row vector $\boldsymbol{o}_i$ of O, the per-token SwiGLU projection matrices are also normalized column-wise using BallNorm, which are expressed as $\bar{W}^i_l = \text{BallNorm}(W^i_l), \quad l \in \{\text{up}, \text{gate}, \text{down}\}$. Then, the per-token normalized SwiGLU is formulated as

$$\text{npSwiGLU}(\boldsymbol{o}_i) = \text{Norm}\Bigg(\Big((\beta^i_{\text{up}} \boldsymbol{o}_i \bar{W}^i_{\text{up}}) \odot \text{Swish}\Big(\sqrt{\frac{L}{B}} \beta^i_{\text{gate}} \boldsymbol{o}_i \bar{W}^i_{\text{gate}}\Big)\Big) \bar{W}^i_{\text{down}}\Bigg), \tag{7}$$

where $\beta^i_{\text{up}}, \beta^i_{\text{gate}} \in \mathbb{R}$ are the learnable token-wise parameters, $\bar{W}^i_{\text{up}}, \bar{W}^i_{\text{gate}} \in \mathbb{R}^{B \times nB}$, $\bar{W}^i_{\text{down}} \in \mathbb{R}^{nB \times B}$, and n controls the hidden dimension. The outputs are stacked and normalized row-wise, followed by a learnable residual update $Y = \text{Norm}\big((1-\alpha_2)O + \alpha_2 \text{npSwiGLU}(O)\big)$. Here, $\alpha_1, \alpha_2 \in (0.05, 1)$ are independently learnable, and Y is the complete nUniMixer block output. After the last block, mean pooling and ℓ_2 normalization are applied before the task-specific towers.

5 EXPERIMENTS

In this section, we conduct extensive experiments to compare the performance of the present UniMixer and nUniMixer architectures with existing state-of-the-art (SOTA) approaches and to answer the following questions: **Q1**: Do the scaling efficiency of the UniMixer and nUniMixer architectures outperform the SOTA architecture? **Q2**: How do individual components contribute to performance and how does UniMixer/nUniMixer perform when scaling in depth? **Q3**: Does the nUniMixer further improve the scaling efficiency? **Q4**: When deployed in a real-world online system, does UniMixer/nUniMixer improve business metrics in A/B testing?

5.1 EXPERIMENTAL SETUP

Datasets and Evaluation Metrics. We evaluate UniMixer on four large-scale public datasets spanning short-video recommendation, advertising, and consumer-to-consumer e-commerce: **QK-Video** (Yuan et al., 2022), **KuaiRand** (Gao et al., 2022), **TAAC-25** (Pan et al., 2026), and **MerRec** (Li et al., 2024). Dataset statistics, training hardware, experimental configurations, and model parameter counts are detailed in Appendix H. We follow the UniRank benchmark[1] (Li et al., 2026a), adopting its standardized preprocessing pipelines, dataset-specific chronological splits, and task definitions to maintain consistent experimental conditions across baselines. The input sequence length is fixed at 100 for all models. We report the area under the ROC curve (AUC) for each prediction task. On the other hand, we use the logged data from the real-world training dataset of **the advertising delivery scenario** 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 short-video application on the day following the users' first activation.

Baselines and Experimental Details. We compare the present 2-blocks/4-blocks UniMixer/nUniMixer architectures with the following representative SOTA frameworks, categorized by modeling paradigm. **Attention-Based Architectures**: HiFormer (Gui et al., 2023), and FAT (Yan et al., 2025), which use the field-specific query, key, and value projections to achieve heterogeneous feature interaction. **TokenMixer-Based Framework**: RankMixer (Zhu et al., 2025) and TokenMixer-Large (Jiang et al., 2026), which employ the rule-based token mixing operation to perform the feature interaction. **FM-Based Framework**: Wukong (Zhang et al., 2024), which concatenates the outputs of an FMB and a linear projection layer to upscale the interaction component. For the industrial experiments, training is conducted using a hybrid distributed framework with 40 GPUs. All compared models in these experiments use Adam for both dense and sparse parameters, with a learning rate of 0.001.

5.2 PERFORMANCE COMPARISON (FOR Q1, Q3)

Table 1 presents the test results of SOTA models and UniMixer/nUniMixer on 17 prediction tasks across four datasets. nUniMixer achieves the highest AUC on 15 of 17 tasks, ranking first on all tasks of KuaiRand, TAAC-25, and MerRec. On QK-Video, it achieves the best performance on Click and Like, while remaining competitive on Follow and Share. Moreover, nUniMixer consistently outperforms UniMixer on all evaluated tasks, demonstrating improvements across diverse recommendation scenarios. For the real-world training dataset of the advertising delivery scenario, the SOTA models with approximately 100 million parameters are used to compare with UniMixer and nUniMixer to explore their scaling laws. The heterogeneous attention architecture serves as the reference 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 nUniMixer architectures significantly outperform other SOTA models across multiple metrics. Subsequently, in this advertising delivery scenario, the performance of RankMixer outperforms all other SOTA models except UniMixer/nUniMixer. Therefore, we select the strongest SOTA model together with UniMixer/nUniMixer for a scaling laws comparison. All models are trained on the same dataset with consistent hyperparameters. Their scaling curves and scaling laws with respect to parameters and FLOPs are given in Fig. 3. As observed, the AUC gains

[1] https://github.com/salmon1802/UniRank

over the reference model exhibit approximate power-law trends as the parameter count or FLOPs increase over the evaluated range. Among two constants in the scaling laws given in Fig. 3, UniMixer achieves the largest scaling exponent across parameters and FLOPs while nUniMixer achieves the largest scaling coefficient across parameters and FLOPs.

Table 1: AUC comparison across datasets and recommendation tasks.

Dataset	Task	UniMixer (Ours)	nUniMixer (Ours)	FAT	Wukong	RankMixer	TokenMixer-Large	HiFormer
QK-Video	Click	0.931873	**0.934531**	0.931368	0.931345	0.930605	0.932396	0.931683
	Follow	0.902906	0.916752	0.917657	0.914388	0.914588	**0.917949**	0.917405
	Like	0.942132	**0.947136**	0.943250	0.940855	0.941830	0.945654	0.944688
	Share	0.926674	0.931684	**0.932305**	0.929892	0.929239	0.932093	0.931408
KuaiRand	Click	0.792876	**0.797017**	0.775447	0.776280	0.783116	0.779543	0.763675
	Follow	0.857537	**0.888198**	0.874392	0.872305	0.869366	0.881161	0.872377
	Like	0.919522	**0.929061**	0.920527	0.920364	0.919990	0.921153	0.918469
	Comment	0.874803	**0.899376**	0.883247	0.885276	0.876950	0.891148	0.886132
	Forward	0.838022	**0.872750**	0.847642	0.840819	0.840369	0.858222	0.855917
	Long View	0.810036	**0.814021**	0.789963	0.794329	0.802038	0.797051	0.778618
TAAC-25	Click	0.812339	**0.834589**	0.824787	0.822436	0.809599	0.831736	0.802022
	Conversion	0.870638	**0.885060**	0.879226	0.877973	0.866537	0.880330	0.871105
MerRec	Like	0.760663	**0.768491**	0.733090	0.734185	0.732832	0.739878	0.739760
	Cart	0.817658	**0.820910**	0.794246	0.798761	0.795891	0.808789	0.801071
	Offer	0.735669	**0.782064**	0.746117	0.745974	0.719009	0.773808	0.757520
	Checkout	0.825165	**0.855737**	0.827274	0.820432	0.831190	0.851837	0.820449
	Purchase	0.826635	**0.855720**	0.818431	0.818980	0.833530	0.853785	0.817747

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

Model	User Retention				Efficiency	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
Heterogeneous Attention (Gui et al., 2023)	0.744577	–	0.733829	–	132.7M	1.68T
HiFormer (Gui et al., 2023)	0.741685	-0.2892%	0.731086	-0.2743%	107.5M	1.37T
Wukong (Zhang et al., 2024)	0.744477	-0.0100%	0.733849	0.0020%	107.1M	1.40T
FAT (Yan et al., 2025)	0.744883	0.0306%	0.734280	0.0451%	138.4M	1.83T
RankMixer (Zhu et al., 2025)	0.749329	0.4752%	0.738938	0.5109%	135.5M	1.68T
TokenMixer-Large (Jiang et al., 2026)	0.748410	0.3833%	0.737940	0.4111%	103.3M	1.27T
UniMixer-2-Blocks (Ours)	0.751121	0.6544%	0.740739	0.6910%	42.4M	2.17T
UniMixer-4-Blocks (Ours)	0.752718	0.8141%	0.742530	0.8701%	84.5M	4.24T
nUniMixer-2-Blocks (Ours)	0.752018	0.7441%	0.741002	0.7173%	51.8M	0.76T
nUniMixer-4-Blocks (Ours)	**0.754199**	**0.9622%**	**0.743929**	**1.010%**	103.2M	1.51T

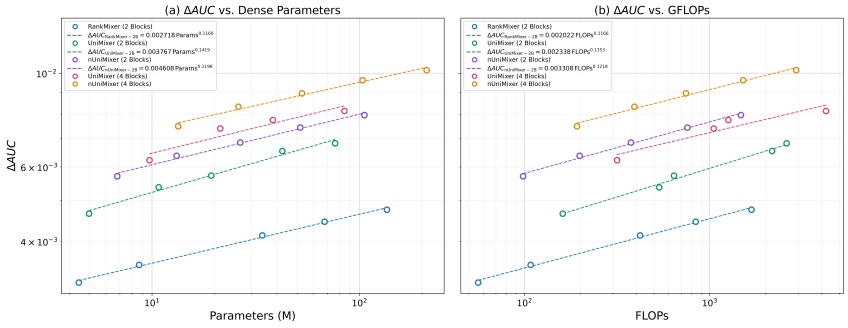


Figure 3: The scaling laws between ΔAUC and Parameters/FLOPs for UniMixer/nUniMixer-2-Blocks, UniMixer/nUniMixer-4-Blocks, and RankMixer architectures. The x-axis is presented on a logarithmic scale.

5.3 Ablation Studies (for Q2)

To explore the contribution of each module in nUniMixer to AUC gains, we conduct ablation studies on various nUniMixer variants. All variants are trained under similar settings. The results are shown

in Table 3, which illustrates that removing any module or violating any normalization operations leads to performance degradation, the pertoken SwiGLU with BallNorm and the DAHM module having the significant impact on overall performance. Besides, according to Table 4, it can be observed that as the depth of UniMixer/nUniMixer 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 UniMixer/nUniMixer with 2 blocks and 4 blocks are shown in Fig. 3, which implies that scaling along depth is more efficient than scaling along width.

Table 3: Ablation on components of nUniMixer.

Setting	**User Retention**				**Efficiency**	
	AUC↑	ΔAUC↑	UAUC↑	ΔUAUC↑	Params	FLOPs/Batch
nUniMixer	0.750446	–	0.739548	–	6.8M	0.098T
w/o DAHM	0.749208	-0.1238%	0.738946	-0.0602%	7.8M	0.098T
w/o BallNorm in nDAHM	0.749255	-0.1191%	0.738954	-0.0594%	6.8M	0.098T
w/o BallNorm in pertoken SwiGLU	0.747673	-0.2773%	0.737226	-0.2322%	6.8M	0.098T
learnable residual update → RMSNorm	0.750169	-0.0277%	0.739900	$+0.0352\%$	6.8M	0.098T

Table 4: Effects of the block number in RankMixer/UniMixer/nUniMixer.

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-2-Blocks	0.749228	–	0.738776	–	4.9675M	0.161T
UniMixer-4-Blocks	0.750803	0.1575%	0.740594	0.1818%	9.715M	0.316T
UniMixer-8-Blocks	0.750875	0.1647%	0.740602	0.1826%	19.2125M	0.629T
nUniMixer-2-Blocks	0.750446	–	0.739548	–	6.8M	0.098T
nUniMixer-4-Blocks	0.752068	0.1622%	0.742135	0.2587%	13.4M	0.191T
nUniMixer-8-Blocks	0.753193	0.2747%	0.743171	0.3623%	20.1M	0.284T

5.4 Online A/B Test Results (for Q4)

To verify the online performance of the proposed UniMixer architecture, we have deployed UniMixer and nUniMixer across multiple advertising delivery scenarios. 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. Besides, the nUniMixer architecture has been fully rolled out across multiple advertising delivery scenarios.

6 Conclusions

In this work, a unified scaling framework is established for scaling laws in recommendation systems, which bridges 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 nUniMixer achieved the best parameter efficiency and computational efficiency. We have deployed and fully rolled out the architectures across multiple advertising delivery scenarios, 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.

AI use statement

In this work, we used generative AI tools for aiding and polishing writing. We have not used generative AI tools for model design, theoretical derivation, code implementation, figure generation.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

We have taken several measures to facilitate the reproducibility of our results. All experiments on public datasets follow the open-source implementation and experimental protocol of UniRank (Li et al., 2026a), including the data preprocessing procedures, dataset splits, feature construction, evaluation metrics, and baseline configurations, unless otherwise specified. Detailed descriptions of the datasets, implementation settings, hyperparameter configurations, and computational resources are provided in Section 5 and Appendix H. We also provide detailed computational procedures for the proposed UniMixer and nUniMixer. The assumptions and complete derivations of our theoretical results are presented in the Appendix D and E. We will release the source code and configuration files upon publication to enable full reproduction of the reported experimental results.

References

Xiaodong Chen, Mingming Ha, Zhenzhong Lan, Jing Zhang, and Jianguo Li. Mobe: Mixture-of-basis-experts for compressing moe-based llms. In *International Conference on Learning Representations*, volume 2026, pp. 40010–40028, 2026.

Qin Ding, Kevin Course, Linjian Ma, Jianhui Sun, Ruochen Liu, Zhao Zhu, Chunxing Yin, Wei Li, Dai Li, Yu Shi, et al. Bending the scaling law curve in large-scale recommendation systems. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7162–7173, 2026.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. Kuairand: An unbiased sequential recommendation dataset with randomly exposed videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, CIKM '22, pp. 3953âĂŞ3957, 2022. doi: 10.1145/3511808.3557624. URL https://doi.org/10.1145/3511808.3557624.

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.

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*, pp. 1725–1731, 2017.

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.

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. ISBN 9781713871088.

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. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7379–7390, 2026.

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.

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. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7489–7500, 2026.

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*, pp. 43–50, 2016.

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.

Honghao Li, Xianquan Wang, Zibin Zhang, Yi Zhang, Kangyi Lin, and Yiwen Zhang. UniRank: Benchmarking Ranking Models for Unified Sequential Modeling and Feature Interaction. *arXiv preprint arXiv:2607.19987*, 2026a.

Lichi Li, Zainul Abi Din, Zhen Tan, Sam London, Tianlong Chen, and Ajay Daptardar. MerRec: A Large-Scale Multipurpose Mercari Dataset for Consumer-to-Consumer Recommendation Systems. *arXiv preprint arXiv:2402.14230*, 2024.

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, 2026b. URL https://arxiv.org/abs/2602.08064.

Ilya Loshchilov, Cheng-Ping Hsieh, Simeng Sun, and Boris Ginsburg. nGPT: Normalized transformer with representation learning on the hypersphere. In *The Thirteenth International Conference on Learning Representations*, 2025.

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, pp. 2022–2031, New York, NY, USA, 2025. Association for Computing Machinery.

Junwei Pan, Wei Xue, Chao Zhou, et al. Tencent Advertising Algorithm Challenge 2025: All-Modality Generative Recommendation. *arXiv preprint arXiv:2604.04976*, 2026.

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. URL https://arxiv.org/abs/2510.15286.

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

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*, pp. 1161–1170, 2019.

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.

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

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *Proceedings of the Web Conference 2021*, pp. 1785–1797, 2021.

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.

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, pp. 6193–6200. Association for Computing Machinery, 2025. ISBN 9798400720406.

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.

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.

Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, Yu Xu, and Xiaohu Qie. TenRec: A Large-Scale Multipurpose Benchmark Dataset for Recommender Systems. *arXiv preprint arXiv:2210.10629*, 2022.

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.

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.

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.

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*, pp. 6309–6316, 2025.

A 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 attention-based architecture 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 Gui et al. (2023), 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 (Yan et al., 2025). HHFT further validates these scaling properties by interleaving heterogeneous Transformer blocks (for preserving domain-specific semantics) with HiFormer blocks (for high-order interaction learning) (Yu et al., 2025). Furthermore, in dynamic user behavior modeling, methods such as HSTU-V1/V2 (Zhai et al., 2024; Ding et al., 2026), MARM (Lv et al., 2025), OneTrans (Zhang et al., 2025), Climber (Xu et al., 2025), Hyformer (Huang et al., 2026), and LLaTTE (Xiong et al., 2026) leverage attention mechanisms to capture long-range temporal dependencies.

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 (Tolstikhin et al., 2021) in computer vision, a paradigm shift towards token-mixing architectures has emerged in industrial recommender systems, yielding advanced models such as RankMixer (Zhu et al., 2025), and TokenMixer-Large (Jiang et al., 2026). RankMixer replaces dynamic attention with static, non-parametric token-mixing operations, achieving competitive CTR prediction performance while maintaining strictly comparable FLOPs (Zhu et al., 2025). Building upon this, TokenMixer-Large (Jiang et al., 2026), Lemur (Han et al., 2025), and MTmixAtt (Qi et al., 2025) design scenario-specific frameworks to fully exploit the scaling efficiency brought by TokenMixer. Nonetheless, a critical gap remains: the design of current token-mixing operators heavily relies on empirical rules and lacks learnability and scenario adaptability.

FM-Based Framework. The FM-based method employs the low-order pairwise modeling for feature interactions in recommendation systems (Rendle, 2010), which was subsequently generalized by Field-aware FMs to capture field-specific and context-sensitive interactions (Juan et al., 2016). 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 extensions, such as DeepFM (Guo et al., 2017), AutoInt (Song et al., 2019), and DCN series (Wang et al., 2017; 2021), integrate MLP or transformer attention to capture high-order interactions. More recently, Wukong (Zhang et al., 2024) 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 (Kaplan et al., 2020; Hoffmann et al., 2022).

B Three Mainstream Scaling Paradigms in Recommendation Systems

Heterogeneous Attention Layer. 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

$$Q_h = \begin{bmatrix} \boldsymbol{x}_1 W_Q^{1h} \\ \vdots \\ \boldsymbol{x}_T W_Q^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, K_h = \begin{bmatrix} \boldsymbol{x}_1 W_K^{1h} \\ \vdots \\ \boldsymbol{x}_T W_K^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, V_h = \begin{bmatrix} \boldsymbol{x}_1 W_V^{1h} \\ \vdots \\ \boldsymbol{x}_T W_V^{Th} \end{bmatrix} \in \mathbb{R}^{T \times d}, \quad (8)$$

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

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

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. 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.

$$\Big[\boldsymbol{x}_t^{(1)} | \, \boldsymbol{x}_t^{(2)} \mid \dots \mid \boldsymbol{x}_t^{(H)}\Big] = \text{SplitHead}(\boldsymbol{x}_t). \quad (9)$$

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

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

The output of TokenMixer is formulated as

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

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

Wukong. In Wukong (Zhang et al., 2024; Hou et al., 2026), the outputs of a factorization machine block (FMB) and a linear projection layer are concatenated to upscale the interaction component, which is formulated as follows

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

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}.

C A Numerical Example of Equivalent Transformation of TokenMixer

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

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

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

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

The output of TokenMixer can be flatten as a vector

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

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

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

According to (11) and (12), 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 following two small matrices

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

D Heterogeneous Feature Interactions

Theorem 3.1 (Equivalent parameterization of TokenMixer). Let $X \in \mathbb{R}^{T\times D}$ be the input hidden state. If the hidden state is processed by the TokenMixer operation, then there exists a permutation matrix W^{perm} such that

$$\text{TokenMixer}(X) = \text{reshape}(W^{\text{perm}}\text{vec}(X)), \tag{13}$$

where $W^{\text{perm}} \in \mathbb{R}^{TD\times TD}$ is a large permutation matrix.

Proof. Considering the input hidden state X and , we have the dimension of the head $B = D/T$. Denote the element in the i-th row and j-th column of X as X_{ij}. For $\boldsymbol{x}_i^{(j)}$ in (9), which denotes the j-th head of the i-th token, then the k-th element of $\boldsymbol{x}_i^{(j)}$ corresponds to $X_{i,(j-1)B+k}$. On the other hand, $\boldsymbol{x}_i^{(j)}$ in (9), the k-th element of $\boldsymbol{x}_i^{(j)}$ corresponds to the $(i-1)B+k$-th of $\boldsymbol{s}^{(j)}$. Finally, the hidden state processed by the TokenMixer operation satisfies

$$S_{j,(i-1)B+k} = X_{i,(j-1)B+k}, \tag{14}$$

where $i, j \in \{1, \ldots, T\}, k \in \{1, \ldots, B\}$. According to (14), after the hidden states are flattened, the following equation holds.

$$\begin{aligned} \text{vec}(S)_{(j-1)D+(i-1)B+k} &= \text{vec}(X)_{(i-1)D+(j-1)B+k} \\ &= \boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k}\text{vec}(X), \end{aligned} \tag{15}$$

where $\text{vec}(X), \text{vec}(S) \in \mathbb{R}^{TD\times 1}$, $\boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k} \in \mathbb{R}^{1\times TD}$, and the $(i-1)D+(j-1)B+k$-th element of vector $\boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k}$ is 1, while all the other elements are 0. Therefore, we can obtain

$$\begin{aligned} \text{vec}(S) &= \begin{bmatrix} \boldsymbol{w}^{\text{perm}}_1 \\ \vdots \\ \boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k} \\ \vdots \\ \boldsymbol{w}^{\text{perm}}_{TD} \end{bmatrix} \text{vec}(X) \\ &= W^{\text{perm}}\text{vec}(X), \end{aligned} \tag{16}$$

where $W^{\text{perm}} \in \mathbb{R}^{TD\times TD}$. □

Theorem 3.2 (Properties of the permutation matrix W^{perm}). Given the input hidden state X, the number of tokens T, and the dimension of the token D, then the permutation matrix W^{perm} satisfies the following properties:

- **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., $\sum_{p=1}^{TD} w^{\text{perm}}_{pq} = 1, \sum_{q=1}^{TD} w^{\text{perm}}_{pq} = 1$.
- **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 an asymmetric matrix.

Proof. In what follows, the properties of W^{perm} are proved, respectively.

- **Compressibility.** According to (15), the index of the input vector $\text{vec}(X)$ satisfies

$$q = (i-1)D+(j-1)B+k = (i-1)TB+(j-1)B+k = \underbrace{[(i-1)T+(j-1)]}_{\text{Block Index } \ell}B+k, \tag{17}$$

where $i, j \in \{1, \ldots, T\}, k \in \{1, \ldots, B\}, \ell = (i-1)T+(j-1)$ is a linear index obtained by flattening a $T \times T$ matrix in row-major order. On the other hand, the index of the input vector $\text{vec}(S)$ satisfies

$$p = (j-1)D+(i-1)B+k = (j-1)TB+(i-1)B+k = \underbrace{[(j-1)T+(i-1)]}_{\text{Block Index } \ell'}B+k, \tag{18}$$

where $\ell' = (j-1)T+(i-1)$ is also a linear index obtained by flattening a $T \times T$ matrix in row-major order. Therefore, the permutation matrix W^{perm} moves the element at the index q to the index p. Constructing a commutation matrix $G \in \mathbb{R}^{T^2\times T^2}$ and an identity matrix $I \in \mathbb{R}^{B\times B}$, they satisfy $G_{\ell',\ell} = 1$ and $I_{k,k} = 1$. Then, the element $W^{\text{perm}}_{p,q} = G_{\ell',\ell} \otimes I_{k,k}$. Therefore, we have $W^{\text{perm}} = G \otimes I$.

- **Doubly Stochasticity.** According to the proof of Theorem 3.1, the $(i-1)D+(j-1)B+k$-th element of the row vector $\boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k}$ in W^{perm} is 1, while all the other elements are 0. Then, we can obtain $\sum_{q=1}^{TD} w^{\text{perm}}_{pq} = 1$. On the other hand, for $((i-1)D+(j-1)B+k)$-th column of W^{perm}, the index of non-zero element are only in the $((j-1)D+(i-1)B+k)$-th row. Therefore, $\sum_{p=1}^{TD} w^{\text{perm}}_{pq} = 1$ holds.

- **Sparsity.** As mentioned in the proof of Theorem 3.1, the $(i-1)D+(j-1)B+k$-th element of the row vector $\boldsymbol{w}^{\text{perm}}_{(j-1)D+(i-1)B+k}$ in W^{perm} is 1, while all the other elements are 0. Therefore, the indices of the elements equal to 1 in the matrix W^{perm} are $\big[(j-1)D+(i-1)B+k, (i-1)D+(j-1)B+k\big]$, where $i, j \in \{1, \ldots, T\}, k \in \{1, \ldots, B\}$. For given i, j, and k, $(j-1)D+(i-1)B+k$ and $(i-1)D+(j-1)B+k$ are in one-to-one correspondence. It implies that each row/column of W^{perm} contains exactly one non-zero element.

- **Symmetry.** $W^{\text{perm}} \in \mathbb{R}^{TD\times TD}$ is a square matrix. According to the indices of non-zero elements $\big[(j-1)D+(i-1)B+k, (i-1)D+(j-1)B+k\big]$ in W^{perm}, for $((i-1)D+(j-1)B+k)$-th row of W^{perm}, the corresponding index of non-zero element is $(j-1)D+(i-1)B+k$, which demonstrates

  $$W^{\text{perm}}_{(j-1)D+(i-1)B+k,(i-1)D+(j-1)B+k} = W^{\text{perm}}_{(i-1)D+(j-1)B+k,(j-1)D+(i-1)B+k}.$$

  Therefore, the permutation matrix W^{perm} under $T = H$ is a symmetric matrix. When $T \neq H$, then

  $$W^{\text{perm}}_{(j-1)D+(i-1)B+k,(i-1)D+(j-1)B+k} \neq W^{\text{perm}}_{(i-1)D+(j-1)B+k,(j-1)D+(i-1)B+k}.$$

  Therefore, the permutation matrix W^{perm} under $T \neq H$ is an asymmetric matrix.

The proof is completed. □

The properties of the permutation matrix W^{perm} are summarized as follows.

Properties of The Permutation Matrix W^{perm}.

- **Compressibility.** $W^{\text{perm}} = G \otimes I$.
- **Doubly Stochasticity.** $\sum_{p=1}^{TD} w^{\text{perm}}_{pq} = 1$ and $\sum_{q=1}^{TD} w^{\text{perm}}_{pq} = 1$.
- **Sparsity.** Each row/column of W^{perm} contains exactly one non-zero element.
- **Symmetry.** $W^{\text{perm}} = W^{\text{perm}\mathsf{T}}$ if $T = H$.

E The Computation Pipeline Optimization of The UniMixing Module

Define $W_{\text{Global}} \in \mathbb{R}^{(L/B)\times L/B}$ and $W^i_{\text{Local}} \in \mathbb{R}$ as follows

$$W_{\text{Global}} = \begin{bmatrix} w^G_{(1,1)} & \cdots & w^G_{(1,L/B)} \\ \cdots & \cdots & \cdots \\ w^G_{(L/B,1)} & \cdots & w^G_{(L/B,L/B)} \end{bmatrix}, W^i_{\text{Local}} = \begin{bmatrix} v^i_{(1,1)} & \cdots & v^i_{(1,B)} \\ \cdots & \cdots & \cdots \\ v^i_{(B,1)} & \cdots & v^i_{(B,B)} \end{bmatrix}, \tag{19}$$

where w_{ij} and v_{ij} are the scalars. According to (3), $\text{vec}(X)$ is evenly split into L/B vectors, which is rewritten as

$$\text{vec}(X) = \Big[\boldsymbol{x}_1 |\ \boldsymbol{x}_2 \mid \ \ldots \ \mid \boldsymbol{x}_{\frac{L}{B}}\Big]^{\mathsf{T}}, \tag{20}$$

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

According to the origin expression of UniMing (2), the term $\Big(W_{\text{Global}} \otimes \{W^i_{\text{Local}}\}_{i=1}^{L/B}\Big)\text{vec}(X)$ can be rewritten as

$$\begin{aligned}\Big(W_{\text{Global}} \otimes \{W^i_{\text{Local}}\}_{i=1}^{L/B}\Big)\text{vec}(X) &= \begin{bmatrix} w^G_{(1,1)}W^1_{\text{Local}} & \dots & w^G_{(1,L/B)}W^{L/B}_{\text{Local}} \\ \dots & \dots & \dots \\ w^G_{(L/B,1)}W^1_{\text{Local}} & \dots & w^G_{(L/B,L/B)}W^{L/B}_{\text{Local}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1^\top \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}}^\top \end{bmatrix} \\ &= \begin{bmatrix} w^G_{(1,1)}W^1_{\text{Local}}\boldsymbol{x}_1^\top + \ldots + w^G_{(1,L/B)}W^{L/B}_{\text{Local}}\boldsymbol{x}_{\frac{L}{B}}^\top \\ \dots \\ w^G_{(L/B,1)}W^1_{\text{Local}}\boldsymbol{x}_1^\top + \ldots + w^G_{(L/B,L/B)}W^{L/B}_{\text{Local}}\boldsymbol{x}_{\frac{L}{B}}^\top \end{bmatrix} \in \mathbb{R}^{L\times 1}\end{aligned} \tag{21}$$

On the other hand, we can obtain the following expression

$$\begin{aligned} & W_{\text{Global}}\text{reshape}\Big(\Big[\boldsymbol{x}_1 W^{1\top}_{\text{Local}} \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}}\Big], \frac{L}{B}, B\Big) \\ &= \begin{bmatrix} w^G_{(1,1)} & \dots & w^G_{(1,L/B)} \\ \dots & \dots & \dots \\ w^G_{(L/B,1)} & \dots & w^G_{(L/B,L/B)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 W^{1\top}_{\text{Local}} \\ \vdots \\ \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}} \end{bmatrix} \\ &= \begin{bmatrix} w^G_{(1,1)}\boldsymbol{x}_1 W^{1\top}_{\text{Local}} + \ldots + w^G_{(1,L/B)}\boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}} \\ \dots \\ w^G_{(L/B,1)}\boldsymbol{x}_1 W^{1\top}_{\text{Local}} + \ldots + w^G_{(L/B,L/B)}\boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}} \end{bmatrix} \in \mathbb{R}^{\frac{L}{B}\times B} \end{aligned} \tag{22}$$

The element in ((21)) and the element in (22) satisfy

$$w^G_{(i,1)}W^1_{\text{Local}}\boldsymbol{x}_1^\top + \ldots + w^G_{(i,L/B)}W^{L/B}_{\text{Local}}\boldsymbol{x}_{\frac{L}{B}}^\top = (w^G_{(i,1)}\boldsymbol{x}_1 W^{1\top}_{\text{Local}} + \ldots + w^G_{(i,L/B)}\boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}})^\top, \tag{23}$$

which results in

$$\Big(W_{\text{Global}}\otimes\{W^i_{\text{Local}}\}_{i=1}^{L/B}\Big)\text{vec}(X) = \text{reshape}\Big(W_{\text{Global}}\text{reshape}\Big(\Big[\boldsymbol{x}_1 W^{1\top}_{\text{Local}} \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}\top}_{\text{Local}}\Big], \frac{L}{B}, B\Big), L, 1\Big)$$

Since both W^i_{Local} and $W^{i\top}_{\text{Local}}$ 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

$$\text{UniMixing}(X) = \text{reshape}\Big(W_{\text{Global}}\text{reshape}\Big(\Big[\boldsymbol{x}_1 W^1_{\text{Local}} \;\Big|\; \boldsymbol{x}_2 W^2_{\text{Local}} \;\Big|\; \dots \;\Big|\; \boldsymbol{x}_{\frac{L}{B}} W^{\frac{L}{B}}_{\text{Local}}\Big], \frac{L}{B}, B\Big), 1, L\Big). \tag{24}$$

F The Difference and Connection of Attention-based, TokenMixer-based, FM-based methods

According to the unified theoretical framework (6), the difference and connection of attention-based, TokenMixer-based, FM-based methods are summarized in the following table.

Table 5: The differences of attention-based, TokenMixer-based, FM-based methods under the unified theoretical framework.

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

G Global Element-Wise Mixing Property of DAHM

According to the equivalent parameterization of TokenMixer, the UniMixer and the DAHM module designs, the numerical examples are given in Fig. 4. From the resulting matrix Z, it can be observed that TokenMixer enables inter-block exchange through a global mixing matrix G; UniMixer first performs intra-block mixer using local mixing parameter matrices, followed by block-level routing through a global mixing parameter matrix. In contrast, DAHM enables global element-level exchange without constructing the large permutation projection W^{perm}.

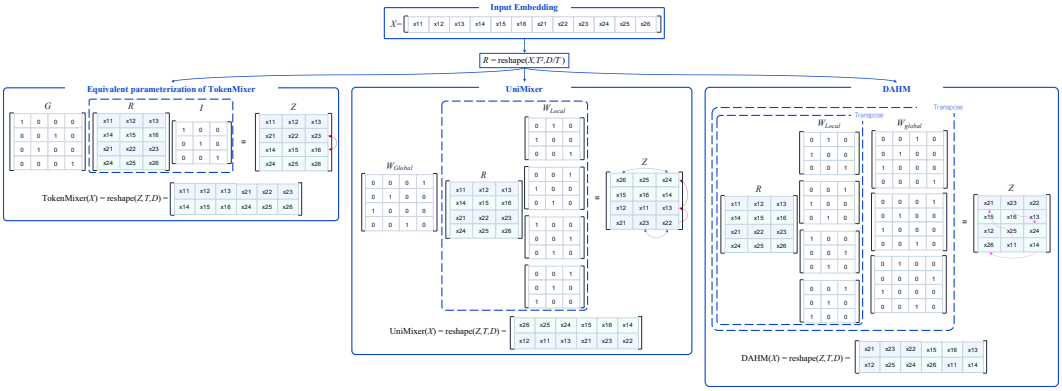


Figure 4: A numerical example of global element-wise mixing property of DAHM.

H Public Dataset Specifications and Experimental Details

Dataset Specifications. Table 6 summarizes the statistics of the four public datasets used in our experiments. M and K denote million and thousand, respectively, and AvgL and MaxL denote the average and maximum behavior sequence lengths.

Table 6: Dataset statistics.

Dataset	Instances	Users	Items	Fields	Tasks	AvgL	MaxL
QK-Video	493.31M	5.00M	3.75M	10	4	99	6.01K
KuaiRand	323.46M	27.29K	32.04M	40	6	11.86K	228.03K
TAAC-25	757.21M	7.71M	15.71M	30	2	98	100
MerRec	172.30M	1.70M	42.58M	20	5	102	26.58K

QK-Video (Yuan et al., 2022) is the short-video subset of TenRec, collected from Tencent's QQ Kandian recommendation platform. It contains user–video impressions with four feedback labels: click, follow, like, and share. User and video side information is also provided. Since timestamps are unavailable, the original interaction order is used to construct user histories.

KuaiRand (Gao et al., 2022) is a short-video recommendation dataset collected from Kuaishou. It combines regular recommendation traffic with randomly exposed videos and provides rich user profiles, video attributes, and timestamped interactions. We evaluate six prediction tasks: click, follow, like, comment, forward, and long view. Its long user histories support the evaluation of sequential recommendation models.

TAAC-25 (Pan et al., 2026) is an industrial advertising dataset released for the Tencent Advertising Algorithm Challenge 2025. It contains anonymized user and item features and three interaction types: exposure, click, and conversion. We evaluate click and conversion prediction, with conversion events also labeled as clicks to preserve the click–conversion relationship.

MerRec (Li et al., 2024) is a consumer-to-consumer recommendation dataset collected from the Mercari marketplace. Following UniRank, we use the October 2023 partition and construct samples anchored at the first item view for each user–item pair. Subsequent interactions provide five prediction targets: like, cart, offer, checkout, and purchase. The dataset also includes product attributes such as category, brand, condition, and price.

For all four datasets, we follow the preprocessing pipelines, dataset-specific chronological splits, and task definitions of UniRank (Li et al., 2026a). Each target interaction is paired with its preceding behavior sequence, and the input sequence length is fixed at 100 for all models.

Model Parameter Counts. Table 7 reports the dense parameter counts of all evaluated models on the four public datasets.

Table 7: Dense parameter counts on the four public datasets. Embedding parameters are excluded, while shared layers and task towers are included. All values are in millions (M).

Model	QK-Video	KuaiRand	TAAC-25	MerRec
UniMixer	12.6944	12.8013	12.5264	12.8105
nUniMixer	12.5095	13.0157	12.7331	12.8048
UniMixer DualBasis	12.6973	12.8041	12.5274	12.8132
FAT	12.6103	12.2130	12.3862	12.9192
Wukong	12.7525	12.5940	12.5980	12.4596
RankMixer	13.0901	12.5893	12.6395	12.6095
TokenMixer-Large	12.6862	12.8924	12.8493	12.8368
HiFormer	12.6924	12.5510	12.6308	12.7181

Training Hardware and Experimental Configurations. All experiments on the public datasets are conducted on two NVIDIA A800 GPUs. Each model is trained for one epoch with a per-GPU batch size of 8192 and two gradient accumulation steps, resulting in an effective global batch size of 32,768. The input sequence length is fixed at 100 for all models.

We use separate optimizers for dense parameters and embeddings. Dense backbone parameters are optimized using AdamW with a learning rate of 10^{-4} and zero weight decay, while embeddings are optimized using Adagrad with a learning rate of 0.05. Task towers also use AdamW with zero weight decay. Their learning rate is set to 10^{-4} for UniMixer, UniMixer DualBasis, and all external baselines, and 2×10^{-3} for nUniMixer. For nUniMixer, full validation is performed every 100 optimizer updates; this affects wall-clock training time but does not change the effective batch size or number of training epochs.

Dense parameter counts, excluding embeddings but including shared layers and task towers, are reported in Table 7.