ELiT: An Expressive Lite Transformer for CTR Prediction with Error-Bounded Attention Approximation

Anonymous authors
Paper under double-blind review

Abstract

Recommender systems (RS) have long been dominated by the Embedding & MLP paradigm, which is now approaching its intrinsic performance ceiling. Motivated by the scaling laws of LLMs, recent studies have begun to adapt Transformer backbones to RS. However, direct transplantation incurs prohibitive parameter overhead, as heterogeneous input features rule out globally shared QKV projections and necessitate token-level parameterization, thereby causing parameter explosion, whereas lightweight variants typically sacrifice global interaction modeling. To resolve this trade-off, we propose the **E**xpressive **Li**te **T**ransformer (**ELiT**), a Transformer variant tailored to heterogeneous features that reconciles efficiency with strong representational power. Our design is motivated by a key empirical insight: in RS, inter-token affinities are governed predominantly by feature fields rather than by specific feature values, rendering the inter-token affinity relationships largely invariant across different inputs and thus permitting aggressive pruning of redundant self-attention (SA) parameters to maintain parameter efficiency. Concretely, ELiT introduces: (i) a progressive two-stage token mixing mechanism, fortified by a sparse MoE module, that substitutes for SA in modeling inter-token relations, thereby achieving sample-adaptive global interaction with negligible overhead; (ii) theoretical guarantees that the approximation error of the attention matrix remains tightly bounded, rigorously justifying its lightweight yet highly expressive architecture; (iii) a masked SwiGLU module that leverages the parameters freed by the simplified SA in (i) to upgrade the standard FFN for higher representational capacity. To avoid the substantial inference overhead of vanilla SwiGLU's dual weight matrices, we replace them with a shared FP16 matrix and binary masks, alleviating the memory-bandwidth bottleneck at negligible computational cost. Extensive experiments on large-scale industrial and public datasets and online A/B tests consistently show that ELiT outperforms SOTA baselines, achieving an exceptional efficiency–expressiveness trade-off in real-world RS.

1 Introduction

Recommender systems (RS) model user preferences and item attributes for personalized ranking, improving user satisfaction and platform revenue. RS architectures have evolved from shallow statistical models to deep learning–based recommendation models (DLRMs), typically under the Emb & MLP paradigm. This paradigm now faces a performance bottleneck, making it a central challenge to surpass DLRM ceilings while sustaining improvement. Recently, inspired by the success and clear scaling laws of large language models (LLMs) in NLP, academia and industry have begun adapting the Transformer backbone to RS, aiming for comparable scaling-law gains in RS.

Recent practical approaches fall into two categories. The first, exemplified by FAT Yan et al. (2025) and HHFT Yu et al. (2025), directly ports Transformers to RS. Since RS input features are inherently heterogeneous—unlike in NLP—these methods replace the globally shared QKV projector with token-wise independent parameters. This raises the dense parameter cost of SA, inevitably hindering online deployment. Later methods, such as RankMixer Zhu et al. (2025), focus on developing lightweight Transformer variants to preserve online serving feasibility. These approaches drastically simplify the self-attention module to remove redundant parameters. However, they lack the ability to capture global feature interactions, which restricts their expressiveness.

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.

To address these issues, we propose ELiT, an expressive lite Transformer variant for heterogeneous features with error-bounded attention approximation. Our design is motivated by a key observation: in RS, inter-token relations exhibit little variation across different inputs, whereas in NLP they vary substantially. This likely stems from the fact that each token encapsulates multiple heterogeneous feature fields; consequently, inter-token affinity is determined mainly by which fields the tokens encode rather than by their specific values, rendering the token–token relations largely input-invariant. This property enables us to prune redundant SA parameters and reallocate the savings to later layers, improving token processing. We summarize ELiT's innovations as follows.

- We introduce a progressive local-to-global two-stage token mixing strategy, enhanced with a sparse MoE for adaptive personalization. This module is designed as a direct counterpart to the self-attention mechanism in Transformers: it performs global token interactions with comparable functionality, yet at a fraction of the cost. The replacement is particularly justified in our setting, where inter-token affinities are largely governed by feature fields rather than specific values—allowing us to learn a universal mixing pattern with negligible loss of expressiveness, while the sparse MoE further captures user-specific deviations with minimal overhead.
- We theoretically establish that the approximation error of the token attention matrix in ELiT remains tightly bounded under the RS setting, despite using substantially fewer parameters than self-attention. This bounded-error guarantee theoretically verifies that ELiT possesses the dual advantages of lightweight design and strong expressiveness, confirming its customized architecture and parameter allocation tailored for RS applications.
- Leveraging the parameter savings from our token-mixing design, we reallocate the freed budget to upgrade the standard FFN to a more expressive gated linear unit (GLU), specifically SwiGLU. Although vanilla SwiGLU boosts representational capacity, its dual weight matrices incur considerable inference overhead. We therefore replace them with a single FP16 matrix and binary masks (1 bit each), preserving SwiGLU's expressiveness while minimizing storage and easing inference. Extensive experiments on industrial and public datasets, together with online A/B tests, validate ELiT's effectiveness and superiority over SOTA methods.

2 Related Work

Recommender systems have evolved from shallow models (e.g., LR Wang et al. (2016), GBDT Friedman (2001)) to deep learning approaches (e.g., DeepFM Guo et al. (2017), DIN Zhou et al. (2018)) that learn embeddings end-to-end. Recently, driven by LLMs and scaling laws Naveed et al. (2025); Bi et al. (2024), researchers have adapted Transformers to recommendation to overcome DLRM bottlenecks. For example, OneTrans Zhang et al. (2026), FAT Yan et al. (2025), and MixFormer Huang et al. (2026) adopt the Transformer for RS but replace the global QKV projection with field-specific parameters to handle feature heterogeneity. This modification, however, inflates self-attention complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(Nd^2)$, where N denotes the number of tokens. Given RS's stringent latency constraints, such heavy architectures are impractical for online deployment.

To address this issue, recent works such as RankMixer Zhu et al. (2025), TokenMixer Jiang et al. (2026), and UniMixer Ha et al. (2026) simplify the self-attention (SA) module to eliminate redundant parameters, ensuring a lightweight design. For example, RankMixer, inspired by MLP-Mixer Tolstikhin et al. (2021), uses fixed, rule-based token mixing over predefined token heads. Being non-parametric, it is inherently lightweight and readily deployable online. However, unlike SA, this rule-based strategy impedes cross-token information exchange and fails to capture global feature interactions. Thus, while offering moderate representational capacity, it severely constrains scalability. Therefore, designing a Transformer variant that balances expressiveness and lightweight deployment under heterogeneous features remains challenging.

3 Methodology

3.1 Overall Architecture

The overall architecture of ELiT is illustrated in Figure 1 and can be roughly divided into four parts: (1) Tokenization Layer (2) LP (Learnable & Personalized) Token Mixing Layer (3) Per-Token

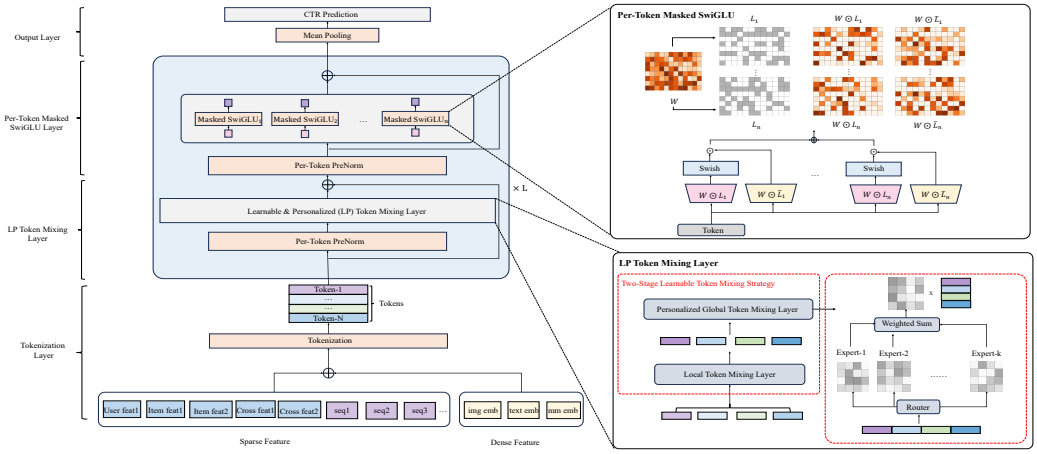


Figure 1: The overview of the proposed ELiT.

Masked SwiGLU Layer, and (4) Output Layer. This architecture can be viewed as a lightweight Transformer variant specifically designed for heterogeneous features. Specifically, assume the input sample comprises multiple features $F_i,\ i \in \{1, \ldots, k\}$. These features are embedded into corresponding embeddings e_i of varying dimensions d_i, i.e., $e_i = \text{Embedding}(F_i, d_i) \in \mathbb{R}^{d_i}$. Following pioneering works like RankMixerZhu et al. (2025) and TokenMixer-LargeJiang et al. (2026), the Tokenization Layer first partitions these raw features into several groups $\{G_1, G_2, \ldots, G_N\}$ based on their semantic similarity, and subsequently applies distinct MLP mappings to compress and align these groups while preserving their heterogeneity, ultimately yielding the final token representations $\{T_1, T_2, \ldots, T_N\}$, where $T_i = \text{MLP}_i(G_i)$. Building upon this, our main innovations lie in the LP Token Mixing Layer and the Per-Token Masked SwiGLU Layer. These components are designed to emulate the key modules of Transformer architectures Vaswani et al. (2017), i.e., the self-attention and feed-forward network (FFN), while simultaneously ensuring strong model expressiveness and effectively controlling both parameter scale and computational cost. Finally, the Output Layer is responsible for performing downstream tasks, such as CTR/CVR prediction Chen et al. (2025).

3.2 LP Token Mixing Layer

Learnable Token Mixing: In the standard Transformer, self-attention captures global context by aggregating token information via pairwise similarities. In RS, however, tokens are highly heterogeneous—coming from different feature fields with distinct semantics—so a single globally shared QK projection is insufficient. Token-level projections are therefore required, which drastically inflates QK parameter overhead and makes deployment in latency-sensitive systems impractical. RankMixer addresses this with a rule-based, parameter-free token mixing strategy. Nonetheless, this heuristic design restricts information flow to a limited number of predefined heads, thereby limiting global context integration. To break this isolation and enable unrestricted cross-token information flow, a natural solution is to learn a recombination matrix that explicitly parameterizes the mixing process. In particular, for an input token matrix $T \in \mathbb{R}^{N \times D}$, one can apply a recombination matrix $W^{\text{perm}} \in \mathbb{R}^{ND \times ND}$ to produce a transformed representation T_{new}, as shown in Eq (1). Unfortunately, this naive approach is impractical for RS: W^{perm} contains $(ND)^2$ parameters, which is prohibitively large, and the excessively enlarged parameter space further poses severe optimization difficulties, impeding both effective learning and stable convergence.

$$T_{\text{new}} = \text{reshape}(W^{\text{perm}} \cdot \text{flatten}(T),\ (N, D)) \tag{1}$$

Motivated by the practical constraints of online deployment and the need for lightweight architectures, we investigate effective ways to compress the global direct fusion strategy. The core challenge lies in the prohibitively large recombination matrix, which necessitates a principled approach to parameter reduction. We therefore adopt Kronecker product decomposition Regalia & Sanjit (1989), factorizing the original large matrix into two smaller matrices G and M, whose sizes are governed by a predefined constant B, as detailed below. In this way, the number of parameters in W^{perm} drops from $(ND)^2$ to $(ND/B)^2 + B^2$, and the resulting T_{new} is formulated as in Eq (3).

$$W^{\text{perm}} = G \otimes M, \quad G \in \mathbb{R}^{\frac{ND}{B} \times \frac{ND}{B}}, \quad M \in \mathbb{R}^{B \times B}, \tag{2}$$

$$T_{\text{new}} = \text{reshape}([G \otimes M] \cdot \text{flatten}(T), N, D) \tag{3}$$

However, naively following the above computation order explicitly materializes the intermediate variable $G \otimes M$, which still demands prohibitive memory. The root cause is that the Kronecker product of the two small matrices G and M is computed first, producing a large intermediate matrix. To circumvent this, we reorder the computation and present an equivalent formulation below. Here, x_i denotes the i-th block obtained by flattening the input token matrix into a vector and splitting it into (ND/B) blocks, each of size B, with $L = ND$ denoting the total flattened dimension. Consequently, the computation can be rewritten as follows.

$$x_i = \text{flatten}(x)\big[\, iB : (i+1)B \,\big], \tag{4}$$

$$T_{\text{new}} = \text{reshape}\Big(G \cdot \text{reshape}\big([\, x_1 M^\top, \ldots, x_{L/B} M^\top \,],\ (\frac{L}{B},\ B)\big),\ (N,\ D)\Big). \tag{5}$$

Eq (5) is equivalent to Eq (3), and the proof of their equivalence is given in Appendix A. By exchanging the computation order, we completely eliminate the need to instantiate the large intermediate matrix. Beyond this memory saving, the new order offers an insightful perspective on the fusion mechanism, which we term ***two-stage progressive fusion***: (1) **Local Fusion:** after flattening the input tokens, we partition them into L/B blocks of size B and perform local feature mixing within each block using the matrix M; (2) **Global Fusion:** we then aggregate information across all blocks via the matrix G to obtain the final output. In this design, the only parameters incurred are those of the two smaller matrices G and M, amounting to $(L/B)^2 + B^2$.

Given the substantial heterogeneity among tokens—each token is constructed by concatenating the feature group produced by a prior group partition and is then compressed and dimensionally aligned by an independent tokenizer, yielding markedly divergent underlying distributions—we adopt a separate matrix M_i with dedicated parameters for each sub-block, so as to better capture the heterogeneous characteristics across tokens. Consequently, our design introduces L/B independent matrices M_i for local fusion, together with a single global matrix G. The resulting token mixing operation and the associated parameter count are presented below.

$$T_{\text{new}} = \text{reshape}\Big(G \cdot \text{reshape}\big([\, x_1 M_1^\top, \ldots, x_L M_L^\top \,],\ (\frac{L}{B},\ B)\big),\ (N,\ D)\Big), \tag{6}$$

$$\text{Params} = (L/B)^2 + (L/B) \cdot B^2. \tag{7}$$

Personalized Token Mixing: The progressive local-to-global scheme makes Token Mixing learnable and flexible, but fusion patterns are inherently user-dependent, which the current architecture ignores. We therefore personalize the Token Mixing module by modeling both the local mixing pattern M and the global mixing pattern G as functions of the user input x. Under heterogeneous features, the L/B matrices M_i and the matrix G have LB and $(L/B)^2$ parameters, respectively. Extending matrix G to multiple experts adds negligible parameters (since the sub-block dim B is relatively large in RS), whereas M_i is much larger. To preserve lightweight design, we apply personalized modeling only to G via a Sparse MoE formulation Riquelme et al. (2021), which captures personalized user interaction patterns by dynamically activating multiple experts. Specifically, we introduce m experts $\{E_i\}_{i=1}^m$. Given a user feature vector x, a routing network $\text{Router}(\cdot)$ computes a score s_i for each expert and activates the top-k experts with the highest scores. Let $\mathcal{S} = \text{TopK}(\mathbf{s}, k)$ be the index set of selected experts, where $\mathbf{s} = [s_1, \ldots, s_m]$. We define an indicator $g_i = \mathbf{1}[i \in \mathcal{S}]$ and a normalized weight α_i over the selected experts. $G(x)$ is then given by

$$G(x) = \sum_{i=1}^{m} g_i \alpha_i E_i(\mathbf{x}), \quad g_i = \mathbf{1}[i \in \mathcal{S}], \quad \alpha_i = \frac{\exp(s_i)}{\sum_{j=1}^{m} g_j \exp(s_j)}. \tag{8}$$

3.3 Variants of Lite Transformers under Heterogeneous Features

Conceptually, ELiT mirrors the backbone Transformer in LLMs; the two share the same underlying rationale, differing only in how their components are tailored to the heterogeneous features of search and recommendation. Self-attention integrates information across tokens in two steps: W_V applies a local transformation within each token, whereas $QK^\top$ captures inter-token similarities for global integration. ELiT plays these two roles with M_i $(i = 1, \ldots, L/B)$ and G, respectively, as summarized in Table 1. Conceptually, therefore, our method is identical in spirit to the self-attention of the Transformer: both follow a local-then-global fusion paradigm. The differences lie in the following two aspects. **(1) Token-specific local fusion.** Instead of applying a single globally shared matrix M, we transform each sub-block with its own independent matrix M_i. This is

Table 1: The relationship between the Self-Attention and LP Token Mixing Layer

	Local Fusion Pattern	Global Fusion Pattern
Self-Attention	$V = XW_V$	$\mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)$
LP Token Mixing Layer	$\mathrm{Concat}\left(X_i M_i\right)$	$G(X)$

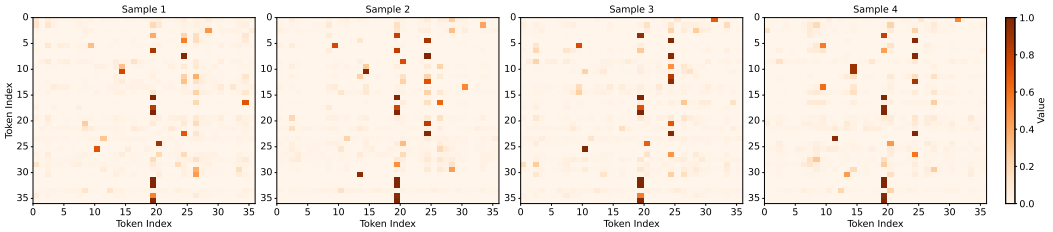


Figure 2: Token association patterns across different samples

because tokens in search and recommendation are intrinsically heterogeneous—a fundamental departure from NLP—and therefore cannot be mapped by a unified linear transformation. **(2) Directly learned global fusion.** Rather than characterizing token similarity via Q/K projections followed by an inner product, we learn the mixing matrix $G(x)$ directly. This preserves the lightweight nature of the model by eliminating the redundant Q/K parameters; indeed, under heterogeneous features the Q/K parameters would have to differ across various tokens, incurring a prohibitive parameter blow-up.

Although the absence of learnable Q/K parameters appears to weaken the model's capacity to express complex relations, we argue that in RS, such a direct transplant of the NLP formulation is parameter-redundant. Concretely, we posit that $G(x)$ demonstrates a high degree of similarity across different users, differing only marginally (illustrated in Figure 2), and thus heavyweight Q/K projections are not needed to capture it. The reason is as follows. In our setting, a token packs together several heterogeneous feature fields (e.g., `item_id`, `query_id`), and the affinity between two tokens is largely determined by which fields they encode rather than by the specific values those fields take. In the search domain, for instance, the interaction between `item_id` and `query_id` is bound to be one of the most salient feature combinations, so the corresponding two tokens are inherently strongly correlated—almost regardless of the particular values of `query_id` and `item_id`. The ideal $G(x)$ can therefore be regarded as a low-variance distribution, expressible as the superposition of a universal pattern (its expectation) and a personalized deviation. This motivates us to model the mixing matrix with a Sparse MoE, in which several activated experts capture its transformation pattern in a personalized manner. In the following, we prove rigorously that, provided the ground-truth $G^*(x)$ has bounded variance, fitting it with k activated routed experts under the SMoE formulation has a bounded approximation error with high probability. The proof proceeds as follows.

Symbol Explanation. Let N be the number of tokens, and let $G^*(x) \in \mathbb{R}^{N\times N}$ be the ground truth of correlation matrix. Let $\bar{G} = \mathbb{E}_x[G^*(x)]$ be the global mean matrix, and let $R(x) = G^*(x) - \bar{G}$ be the residual matrix. We use $\|\cdot\|_F$ to denote the Frobenius norm. Let k be the number of activated experts, and let $\{E_1, \ldots, E_k\} \in \mathbb{R}^{N\times N}$ be the activated expert matrices. The MoE output is defined as the linear combination $G_{\mathrm{MoE}}(x) = \sum_{i=1}^{k} w_i(x) E_i$. Let $V = \mathrm{span}\{E_1, \ldots, E_k\}$, so $\dim V \le k$. Let $U \subseteq \mathbb{R}^{N\times N}$ be the low-dimensional residual subspace with $r = \dim U$, and assume $\bar{G} \in U$. Let $\delta > 0$ be an upper bound on the variance of the residual, and let $\alpha \in (0, 1)$ be the confidence parameter. Define the radius bound $\rho = \sqrt{\delta/\alpha}$.

Assumption 1 (Bounded variance of the residual)**.** There exists a constant $\delta > 0$ such that

$$\mathbb{E}_x\left[\|G^*(x) - \bar{G}\|_F^2\right] = \mathbb{E}_x\left[\|R(x)\|_F^2\right] \le \delta.$$

Assumption 2 (Low-dimensional residual subspace)**.** There exist an integer $r \ll N^2$ and an r-dimensional linear subspace $U \subseteq \mathbb{R}^{N\times N}$ such that $\bar{G} \in U$ and, with probability at least $1 - \alpha$,

$$R(x) \in U.$$

Theorem 1. *Let U be the r-dimensional residual subspace and let the MoE use k experts $\{E_1, \ldots, E_k\}$. Then the approximation error satisfies the following relationship. When $r \le k$, there exist expert matrices $\{E_1, \ldots, E_k\}$ such that the MoE output exactly reproduces the ground*

truth. When $r > k$, any choice of k experts cannot cover the entire r-dimensional residual subspace, and the worst-case error is $\sqrt{\delta/\alpha}$ with probability $1 - \alpha$.

$$\boxed{\|G_{\mathrm{MoE}}(x) - G^*(x)\|_F = \begin{cases} 0, & r \le k, \\ \sqrt{\dfrac{\delta}{\alpha}}, & r > k. \end{cases}}$$

Proof. **Case 1: $r \le k$.** Since $r = \dim U \le k$, take an orthonormal basis $\{U_1, \dots, U_r\}$ of U; since $\bar{G} \in U$, write $\bar{G} = \sum_{i=1}^{r} a_i U_i$. Set $E_i = U_i \, \{i = 1, \dots, r\}$ and $E_i = 0 \, \{i = r+1, \dots, k\}$. Then $V = \mathrm{span}\{E_1, \dots, E_k\} = \mathrm{span}\{U_1, \dots, U_r\} = U$, so $U \subseteq V$ and $\bar{G} \in V$. For any $R(x) \in U$, write $R(x) = \sum_{i=1}^{r} c_i(x) U_i$. Since $G^*(x) = \bar{G} + R(x)$ and both $\bar{G}$ and $R(x)$ lie in V, we have $G^*(x) \in V$. Hence there exist weights $\{w_1(x), \dots, w_k(x)\}$ such that $G_{\mathrm{MoE}}(x) = \sum_{i=1}^{k} w_i(x) E_i = G^*(x)$, and therefore $\|G_{\mathrm{MoE}}(x) - G^*(x)\|_F = 0$. **Case 2:$r > k$.** By Assumption 1 and Markov's inequality, we have $\Pr(\|R(x)\|_F > R) \le \mathbb{E}_x[\|R(x)\|_F^2]/R^2 \le \delta/R^2$; setting $R = \sqrt{\delta/\alpha}$ yields $\Pr(\|R(x)\|_F \le \sqrt{\delta/\alpha}) \ge 1 - \alpha$. Hence, with probability at least $1 - \alpha$, $R(x) \in U$ and $\|R(x)\|_F \le \rho$, where $\rho = \sqrt{\delta/\alpha}$. Fix any activated experts $\{E_1, \dots, E_k\}$ and let $V = \mathrm{span}\{E_1, \dots, E_k\}$, so $\dim V \le k$. Since $r = \dim U > k \ge \dim V$, V cannot contain the whole U; hence there exists a nonzero vector $z \in U \cap V^{\perp}$ with $\|z\|_F = 1$. Take $R = \rho z \in U$, so $\|R\|_F = \rho$. For any $V' \in V$, since $z \perp V'$, we have $\|R - V'\|_F^2 = \|\rho z - V'\|_F^2 = \rho^2 \|z\|_F^2 + \|V'\|_F^2 = \rho^2 + \|V'\|_F^2 \ge \rho^2$, and therefore $\inf_{V' \in V} \|R - V'\|_F \ge \rho$. Conversely, for any $R \in U$ with $\|R\|_F \le \rho$, taking $V' = 0 \in V$ gives $\inf_{V' \in V} \|R - V'\|_F \le \|R\|_F \le \rho$. Hence

$$\sup_{R \in U, \, \|R\|_F \le \rho} \inf_{V' \in V} \|R - V'\|_F = \rho = \sqrt{\frac{\delta}{\alpha}}.$$

Correspondingly, for the MoE output, since $G_{\mathrm{MoE}}(x) \in V$ and $G^*(x) = \bar{G} + R(x)$ with $\bar{G} \in U$, we have $\|G_{\mathrm{MoE}}(x) - G^*(x)\|_F = \mathrm{dist}(G^*(x), V)$. Therefore, the worst-case error supremum is

$$\sup_x \|G_{\mathrm{MoE}}(x) - G^*(x)\|_F = \sqrt{\frac{\delta}{\alpha}},$$

and by Assumption 1 and Markov's inequality, with probability at least $1 - \alpha$, $\|G_{\mathrm{MoE}}(x) - G^*(x)\|_F \le \sqrt{\delta/\alpha}$. Moreover, from the construction above, this upper bound is tight and cannot be further reduced. Case 2 is proved. □

In the RS setting, we have $D \gg N$ and $D \gg k$. Even if the number of experts k increases to N^2—which ensures that the theoretical approximation error approaches zero—the parameter consumption remains very small. Specifically, a general heterogeneous Transformer uses $2ND^2$ dense parameters to model cross-token relationships, whereas ELiT uses only $N^2 \cdot (L/B)^2$ dense parameters. In practice, the sub-block dimension B is on the same order of magnitude as D ($B = D/4$ in our experiments), so ELiT requires very few parameters. Consequently, our method can be viewed as an efficient Transformer variant specifically designed for heterogeneous feature scenarios.

3.4 Per-Token Masked SwiGLU

In the standard Transformer architecture, the Feed-Forward Network (FFN) serves as a transformation engine that independently refines each token's representation. To further enhance this capability, the Swish-activated Gated Linear Unit (SwiGLU) Shazeer (2020) has been proposed as an alternative to the FFN, owing to its stronger expressive power. However, this improvement comes at a cost: SwiGLU requires approximately twice as many memory reads as its non-gated counterpart, since it employs separate weight matrices for the gate and value streams. This overhead becomes particularly acute in recommendation scenarios, where systems demand high responsiveness and low latency. In essence, the bottleneck is not computational but memory-bandwidth-limited, stemming from the two independent weight reads inherent to the gating mechanism. Under FP16 precision Haidar et al. (2018), reading the two matrices $W_{up}, W_{gate} \in \mathbb{R}^{h \times d}$ incurs a total of $16 \times 2hd = 32hd$ bits of memory load per token.

An intuitive way to address this issue is to compress the memory footprint of W_{up} and W_{gate}. We retain only a single FP16-precision matrix W and introduce an additional binary mask matrix $L \in \{0,1\}^{h\times d}$. We then generate $W_{up} = W \odot L$ and $W_{gate} = W \odot \bar{L}$, respectively, where $\bar{L} = \mathbf{1} - L$ denotes the complementary mask. Although this scheme still requires storing both W and L, the latter is a Boolean matrix whose entries are restricted to $\{0,1\}$, which significantly reduces the memory cost. Specifically, the total storage footprint drops from $16 \times 2hd$ bits for two FP16 matrices to $(16+1) \times hd$ bits for one FP16 matrix and its accompanying binary mask. Under this setting, the SwiGLU operation can be expressed as follows.

$$\text{SwiGLU}(x) = \text{Swish}(x \odot (W \odot L)) \odot (x \odot (W \odot \bar{L})) \tag{9}$$

However, employing a single binary mask L may unduly restrict the expressive power of W_{up} and W_{gate}. Therefore, we extend the masking scheme to a family of binary masks $\mathbf{L} = \{L_i\}_{i=1}^{n}$, where n denotes the number of binary masks and the Grouped SwiGLU operation is shown as follows.

$$\text{SwiGLU}(x) = \sum_{i=1}^{n} \text{Swish}(x \odot (W \odot L_i)) \odot (x \odot (W \odot \bar{L}_i)) \tag{10}$$

In this case, the memory load is $(16+n)hd$ bits, which is lower than the $32hd$ bits of the standard SwiGLU whenever $n < 16$, alleviating the memory loading burden in the inference stage. However, this extension inevitably incurs additional computational overhead. In particular, while the standard SwiGLU requires $4hd$ FLOPs, the masked variant demands $2nhd$ FLOPs. Consequently, for $n > 2$, the masked approach imposes a considerably larger computational burden. Nevertheless, as a memory-bound operation, the extra computational cost in masked SwiGLU is almost negligible. Overall, the inference latency is reduced. Finally, we extend the Masked SwiGLU to a token-specific formulation, where each token employs an independent set of parameters, thereby better accommodating the heterogeneity across different tokens in RS.

4 Experiments

4.1 Experiment Setting

Datasets Our offline experiments use both industrial and public datasets. The industrial ones come from a large-scale e-commerce platform with 100 million daily active users (DAU), which includes two subsets: D-Search (for search) and D-Rec (for main feed stream). Their daily training volumes are roughly 2.1 billion and 1.4 billion records, with about 650 and 500 features, respectively. For public benchmarks, we additionally adopt two widely used datasets: **Criteo** Criteo Labs (2014), the de-facto CTR benchmark comprising roughly 46 million anonymized ad-click records, and **RecFlow** Liu et al. (2025b), an industrial full-flow dataset released by Kuaishou, of which we use only the *exposure* (impression) subset, i.e., the items actually shown to users at the final stage of the funnel, amounting to about 38 million impressions over 37 days. Since Criteo provides no user grouping, we report only AUC on it, whereas both AUC and GAUC are reported on RecFlow and the industrial datasets. Dataset statistics and the full preprocessing pipeline are detailed in Appendix D.

Evaluation Metrics. **AUC/GAUC**: We use AUC (Area Under the Curve) and GAUC (Group-Level AUC) as the primary effectiveness metrics for model evaluation. **Params**: We use Params to measure the complexity of various models, which represents the number of pure dense parameters. **Training FLOPs per Sample**: Refers to the number of floating-point operations required for the model to process a single sample, representing the computational cost of training.

Compared Methods. We compare the proposed ELiT against the following baselines. **DLRM-MLP** follows the traditional Embedding & MLP paradigm and directly uses an MLP for feature interaction. **GDCN** Wang et al. (2023) proposes a Gated Deep Cross Network to explicitly model high-order feature interactions while dynamically pruning less important cross-terms via an information gate at each order. **RankMixer** Zhu et al. (2025) adopts a rule-based token mixing strategy with a Per-Token FFN for feature interaction. **TokenMixing-Large** Jiang et al. (2026) further extends RankMixer by incorporating a "Mixing & Reverting" mechanism to improve model flexibility. **UniMixer** Ha et al. (2026) improves RankMixer by making the fixed token mixing strategy learnable, thereby boosting the model's expressive ability. **Transformer** directly replicates the self-attention mechanism in recommendation systems to model feature interactions. **FAT** Yan et al. (2025) proposes token-specific Q/K/V projections to accommodate heterogeneous features.

Implementations. All experiments are carried out on a hybrid distributed training system equipped with 128 AMD MI308X GPUs. We adopt Muon Liu et al. (2025a) for dense parameters and AdamW Loshchilov & Hutter (2019) for sparse ones, capitalizing on Muon's superior performance, half cost and memory, and robustness, while retaining AdamW's effectiveness for sparse gradient updates. The batch size and learning rate are uniformly configured as 20,480 and 4×10^{-4} across all experimental settings, and the entire implementation is built upon the PyTorch framework.

4.2 Experimental Results

Comparison with SOTA methods. Tables 2 and 3 summarize the experimental results on the public and industrial datasets, respectively. The corresponding run-to-run standard deviations for the public datasets are reported in Appendix B (Table 5). With a comparable number of parameters, ELiT achieves the best performance on four datasets. In particular, under the real-world industrial setting, ELiT substantially outperforms the DLRM-MLP baseline, yielding AUC and GAUC improvements of 1.07% and 1.55% on D-Search, and 0.14% and 0.42% on D-Rec, respectively.

Table 2: The offline experimental results on public datasets.

Method	Criteo			RecFlow			
	AUC	Params	FLOPs	AUC	GAUC	Params	FLOPs
GDCN	0.7947	100M	200M	0.7296	0.6814	100M	201M
RankMixer	0.8032	95M	191M	0.7388	0.6929	98M	198M
TokenMixer-Large	0.8045	97M	195M	0.7379	0.6927	101M	202M
UniMixer	0.7982	99M	199M	0.7346	0.6888	97M	196M
FAT	0.8016	99M	491M	0.7361	0.6882	99M	277M
Transformer	0.7963	101M	7.74G	0.7326	0.6830	101M	4.34G
ELiT	$\mathbf{0.8065}^*$	101M	210M	$\mathbf{0.7401}^*$	$\mathbf{0.6954}^*$	100M	202M

Note. AUC and GAUC are means over five independent runs with different random seeds. The corresponding standard deviations are reported in Appendix B (Table 5). * denotes a significant improvement over GDCN on the corresponding metric (two-sided Welch's t-test on run-level scores; $p < 0.01$).

Table 3: The offline experimental results on industrial datasets.

Method	D-Search				D-Rec			
	AUC	GAUC	Params	FLOPs	AUC	GAUC	Params	FLOPs
DLRM-MLP	0.7832	0.7285	85M	185M	0.7459	0.6512	90M	181M
GDCN	0.7859	0.7295	82M	183M	0.7481	0.6543	99M	197M
RankMixer	0.7912	0.7375	83M	183M	0.7496	0.6577	94M	188M
TokenMixer-Large	0.7902	0.7364	83M	183M	0.7505	0.6586	93M	186M
UniMixer	0.7899	0.7360	84M	186M	0.7503	0.6583	93M	187M
FAT	0.7891	0.7357	88M	209M	0.7490	0.6573	99M	243M
Transformer	0.7890	0.7355	83M	1.83G	0.7491	0.6579	95M	6.24G
ELiT	**0.7939**	**0.7440**	87M	191M	**0.7519**	**0.6628**	96M	204M

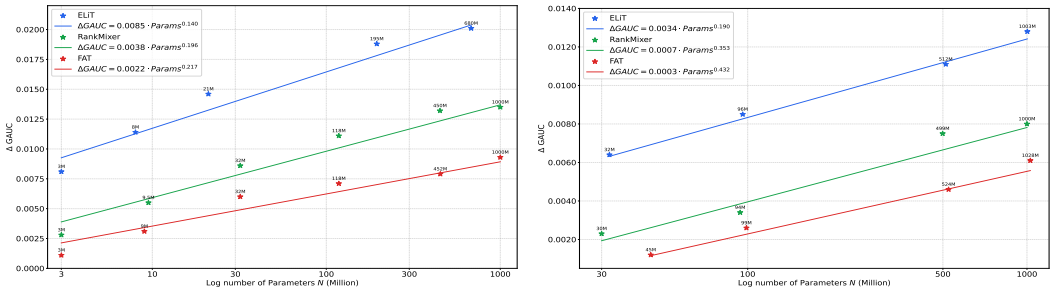
Table 4: Ablation on components of ELiT.

Method	D-Search		D-Rec	
	AUC	GAUC	AUC	GAUC
w/o LTMS	-0.25%	-0.42%	-0.15%	-0.21%
w/o PTMS	-0.21%	-0.32%	-0.11%	-0.14%
w/o MSwiGLU	-0.08%	-0.09%	-0.05%	-0.07%

Ablation Study. We conduct an ablation study to verify the contribution of each module in ELiT. Specifically, we remove (1) the Learnable Token Mixing Strategy (LTMS), (2) the Personalized Token Mixing Strategy (PTMS), and (3) the Masked SwiGLU (MSwiGLU) module, respectively, to observe their individual impacts. The corresponding results are presented in Table 4.

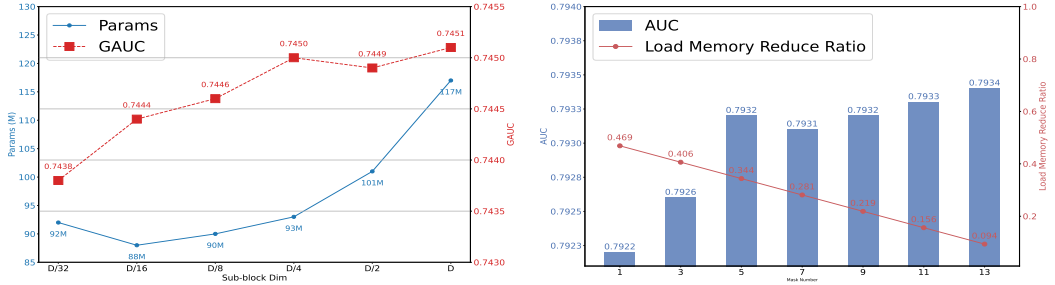
Scaling law. To compare the extensibility of various models, we investigate their scaling laws by scaling the number of dense parameters from small to large and plotting the corresponding AUC gain curves. The results show that ELiT achieves the steepest scaling trajectory among the compared methods, surpassing RankMixer (the most representative lightweight approach) and the heterogeneous Transformer variant FAT. The scaling curves for D-Search and D-Rec are shown in Figure 3.

Online Performance. We deployed ELiT online on both search and main feed stream scenarios. In A/B tests with users randomly assigned via MD5 hashing of device IDs for unbiased traffic allocation, the experimental group (over 20% of traffic) consistently outperformed the baseline. In search, it achieved a 1.76% lift in per capita orders and a 1.32% increase in GMV. In the main feed stream, CTR, per capita orders, and GMV are improved by 1.52%, 2.47%, and 2.18%, respectively.



(a) D-Search (b) D-Rec

Figure 3: Scaling Laws on different datasets.



(a) The selection of sub-block size (b) The number of masked matrices in MSwiGLU

Figure 4: Hyperparameter tuning experiments.

Hyperparameter Tuning. Sub-block size: The sub-block dimension B controls both the interaction range (local/global) and the parameter count. We generally set $B < D$ to avoid disrupting the original token boundaries. A large B sharply increases parameters, whereas a small B restricts local interaction. We therefore vary $B \in \{D/32, D/16, \ldots, D\}$ and evaluate the resulting parameter scale and AUC, as shown in Figure 4a. AUC first rises sharply and then saturates when $B > D/4$, while the parameter count keeps growing rapidly. We thus choose $B = D/4$ to balance the two factors. **The number of masked matrices in MSwiGLU:** Increasing the number of masked matrices improves MSwiGLU's expressiveness but reduces memory savings. As shown in Figure 4b, AUC saturates after $n = 5$, whereas the memory reduction ratio decays linearly. We therefore set $n = 5$.

5 Conclusion

We present ELiT, an expressive lite Transformer for CTR prediction with error-bounded attention approximation. Revisiting self-attention through feature-field dominance, we observe that inter-token affinities in RS are governed mainly by feature fields rather than specific values, enabling aggressive yet theoretically grounded parameter reduction. ELiT introduces a progressive local-to-global token mixing strategy with learnable, personalized components, where a sparse MoE formulation captures user-specific deviations at negligible cost. It further upgrades the FFN with a masked SwiGLU module that preserves representational capacity while substantially reducing memory-bandwidth consumption via binary masks. Theoretically, we derive a rigorous approximation error bound showing that, under the low-variance property of RS, MoE-based mixing faithfully approximates standard QK-based attention with far fewer parameters. Extensive offline and online experiments show that ELiT consistently outperforms state-of-the-art (SOTA) baselines, validating its effectiveness and efficiency for real-world deployment.

AI USE STATEMENT

We used ChatGPT solely to correct grammatical errors, improve sentences, and check for typos. All technical ideas, experimental designs, data analysis, and final content generation were conducted entirely by the human authors. No part of the core research contribution was generated by AI. The authors take full responsibility for the paper.

REFERENCES

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Rong Chen, Shuzhi Cao, Ailong He, Shuguang Han, and Jufeng Chen. Prectr: A synergistic framework for integrating personalized search relevance matching and ctr prediction. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 152–161, 2025.

Criteo Labs. Display advertising challenge (criteo labs). `https://www.kaggle.com/c/criteo-display-ad-challenge`, 2014. Kaggle competition dataset.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

Mingming Ha, Guanchen Wang, Linxun Chen, Xuan Rao, Yuexin Shi, Tianbao Ma, Zhaojie Liu, Yunqian Fan, Zilong Lu, Yanan Niu, et al. Unimixer: A unified architecture for scaling laws in recommendation systems. *arXiv preprint arXiv:2604.00590*, 2026.

Azzam Haidar, Stanimire Tomov, Jack Dongarra, and Nicholas J Higham. Harnessing gpu tensor cores for fast fp16 arithmetic to speed up mixed-precision iterative refinement solvers. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 603–613. IEEE, 2018.

Xu Huang, Hao Zhang, Zhifang Fan, Yunwen Huang, Zhuoxing Wei, Zheng Chai, Jinan Ni, Yuchao Zheng, and Qiwei Chen. Mixformer: Co-scaling up dense and sequence in industrial recommenders. *arXiv preprint arXiv:2602.14110*, 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. *arXiv preprint arXiv:2602.06563*, 2026.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025a.

Qi Liu, Kai Zheng, Rui Huang, Wuchao Li, Kuo Cai, Yuan Chai, Yanan Niu, Yiqun Hui, Bing Han, Na Mou, Hongning Wang, Wentian Bao, Yunen Yu, Guorui Zhou, Han Li, Yang Song, Defu Lian, and Kun Gai. Recflow: An industrial full flow recommendation dataset. In *International Conference on Learning Representations (ICLR)*, 2025b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

Phillip A Regalia and Mitra K Sanjit. Kronecker products, unitary matrices and signal processing applications. *SIAM review*, 31(4):586–613, 1989.

Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in neural information processing systems*, 34:8583–8595, 2021.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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.

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

Fangye Wang, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. Towards deeper, lighter and interpretable cross network for ctr prediction. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 2523–2533, 2023.

Yaozheng Wang, Dawei Feng, Dongsheng Li, Xinyuan Chen, Yunxiang Zhao, and Xin Niu. A mobile recommendation system based on logistic regression and gradient boosting decision trees. In *2016 International joint conference on neural networks (IJCNN)*, pp. 1896–1902. IEEE, 2016.

Bencheng Yan, Yuejie Lei, Zhiyuan Zeng, Zheye Deng, Di Wang, Kaiyi Lin, Pengjie Wang, Chuan Yu, 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.

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. In *Proceedings of the ACM Web Conference 2026*, pp. 8162–8170, 2026.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1059–1068, 2018.

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 THE EQUIVALENCE BETWEEN EQ.5 AND EQ.3

Proof. Define the matrices G and M as follows:

$$G = \begin{bmatrix} w_G^{(1,1)} & \cdots & w_G^{(1,L//B)} \\ \vdots & \ddots & \vdots \\ w_G^{(L//B,1)} & \cdots & w_G^{(L//B,L//B)} \end{bmatrix}, \quad M = \begin{bmatrix} v^{(1,1)} & \cdots & v^{(1,B)} \\ \vdots & \ddots & \vdots \\ v^{(B,1)} & \cdots & v^{(B,B)} \end{bmatrix} \tag{11}$$

and let the flattened input be partitioned into L/B row blocks of length B:

$$\text{flatten}(X) = \left[\, x_1 \mid x_2 \mid \cdots \mid x_{\frac{L}{B}} \,\right]^\top \tag{12}$$

Expanding the product yields

$$\begin{aligned} (G \otimes M)\,\text{flatten}(X) &= \begin{bmatrix} w_G^{(1,1)} M & \cdots & w_G^{(1,L//B)} M \\ \vdots & \ddots & \vdots \\ w_G^{(L//B,1)} M & \cdots & w_G^{(L//B,L//B)} M \end{bmatrix} \begin{bmatrix} x_1^\top \\ \vdots \\ x_{L//B}^\top \end{bmatrix} \\ &= \begin{bmatrix} w_G^{(1,1)} M x_1^\top + \cdots + w_G^{(1,L//B)} M x_{L//B}^\top \\ \vdots \\ w_G^{(L//B,1)} M x_1^\top + \cdots + w_G^{(L//B,L//B)} M x_{L//B}^\top \end{bmatrix} \in \mathbb{R}^{L\times 1}. \end{aligned} \tag{13}$$

Expanding the rewritten expression gives

$$G\,\text{reshape}\Big(\left[\, x_1 M^\top \mid \cdots \mid x_{L//B} M^\top \,\right],\ \frac{L}{B},\ B\Big) \tag{14}$$

$$= \begin{bmatrix} w_G^{(1,1)} & \cdots & w_G^{(1,L//B)} \\ \vdots & \ddots & \vdots \\ w_G^{(L//B,1)} & \cdots & w_G^{(L//B,L//B)} \end{bmatrix} \begin{bmatrix} x_1 M^\top \\ \vdots \\ x_{L//B} M^\top \end{bmatrix} \tag{15}$$

$$= \begin{bmatrix} w_G^{(1,1)} x_1 M^\top + \cdots + w_G^{(1,L//B)} x_{L//B} M^\top \\ \vdots \\ w_G^{(L//B,1)} x_1 M^\top + \cdots + w_G^{(L//B,L//B)} x_{L//B} M^\top \end{bmatrix} \in \mathbb{R}^{\frac{L}{B}\times B}. \tag{16}$$

For every $i = 1, \ldots, L$, the i-th block of the first result and the i-th row of the second result satisfy

$$w_G^{(i,1)} M x_1^T + \cdots + w_G^{(i,L//B)} M x_{L//B}^T = \left(w_G^{(i,1)} x_1 M^\top + \cdots + w_G^{(i,L//B)} x_{L//B} M^\top \right)^\top \tag{17}$$

since $M x_j^\top = \left(x_j M^\top\right)^\top$ for all j. Hence the column vector obtained under the original order is exactly the row-major flattening of the matrix obtained under the exchanged order, and applying $\text{reshape}(\cdot, (N, D))$ to either of them produces the same tensor. Therefore

$$\begin{aligned} T_{\text{new}} &= \text{reshape}\big((G \otimes M)\,\text{flatten}(T),\ (N, D)\big) \\ &= \text{reshape}\Big(G\,\text{reshape}\big([\, x_1 M^\top, \ldots, x_{L/B} M^\top \,],\ (\frac{L}{B}, B)\big),\ (N, D)\Big), \end{aligned} \tag{18}$$

which completes the proof. □

B Run-to-Run Variability on Public Datasets

Table 5 reports the standard deviations corresponding to the public-dataset results in Table 2.

Table 5: AUC and GAUC results with standard deviations on public datasets.

Method	Criteo	RecFlow	
	AUC	AUC	GAUC
GDCN	$0.7947 \pm 1.8 \times 10^{-4}$	$0.7296 \pm 2.9 \times 10^{-4}$	$0.6814 \pm 3.2 \times 10^{-4}$
RankMixer	$0.8032 \pm 2.1 \times 10^{-4}$	$0.7388 \pm 1.2 \times 10^{-4}$	$0.6929 \pm 2.5 \times 10^{-4}$
TokenMixer-Large	$0.8045 \pm 1.4 \times 10^{-4}$	$0.7379 \pm 1.9 \times 10^{-4}$	$0.6927 \pm 3.4 \times 10^{-4}$
UniMixer	$0.7982 \pm 2.3 \times 10^{-4}$	$0.7346 \pm 2.3 \times 10^{-4}$	$0.6888 \pm 3.2 \times 10^{-4}$
FAT	$0.8016 \pm 1.2 \times 10^{-4}$	$0.7361 \pm 1.7 \times 10^{-4}$	$0.6882 \pm 2.9 \times 10^{-4}$
Transformer	$0.7963 \pm 2.1 \times 10^{-4}$	$0.7326 \pm 1.7 \times 10^{-4}$	$0.6830 \pm 2.1 \times 10^{-4}$
ELiT	$0.8065 \pm 1.9 \times 10^{-4}$	$0.7401 \pm 2.1 \times 10^{-4}$	$0.6954 \pm 3.2 \times 10^{-4}$

Note. Values are reported as mean $\pm$ standard deviation over five independent runs with different random seeds.

C Code and Implementation

The source code and implementation details are provided in the supplementary material.

D Dataset Details and Preprocessing

We describe the preprocessing used in all reported experiments. Both public datasets follow the same principle: the train/validation/test split is fixed *first*, all statistics and vocabularies are then fitted on the training split *only*, and the three splits are subsequently transformed by an identical set of rules. We deliberately avoid randomly partitioning already-processed data, so that no information from the evaluation splits can leak into normalization statistics or vocabularies. Categorical ids follow a shared convention on both datasets, namely 0 = PAD, 1 = OOV, 2 = MISSING, and 3 onwards for values observed during training; values that appear only in validation or test are mapped to OOV and never used to extend the vocabulary. The resulting splits are materialized once and shared by all models, so every method consumes exactly the same data version.

Criteo. We use only the labeled file of the Kaggle Display Advertising Challenge ($45{,}840{,}617$ rows) and split it by original row order into $[0,\ 39{,}291{,}957)$ for training, $[39{,}291{,}957,\ 42{,}566{,}287)$ for validation, and $[42{,}566{,}287,\ 45{,}840{,}617)$ for testing, i.e., a ratio of about $12/14 : 1/14 : 1/14$. Although the file is ordered by time, it carries no per-row timestamp, so these three segments should not be interpreted as exact calendar days; the unlabeled `test.txt` is not used as our offline test set. The original click label is kept as-is. Each of the 13 numerical fields is first transformed by $x \mapsto \mathrm{sign}(x) \cdot \log(1 + |x|)$ and then standardized using the mean of the non-missing training values and the corresponding population standard deviation; for every field we additionally store a missing indicator and set the standardized value of a missing entry to 0. For each of the 26 categorical fields we build an independent, complete training-only vocabulary; no low-frequency truncation is applied.

RecFlow. We use the exposure logs (37 daily partitions) and split them by natural date: 2024-01-13 through 2024-02-16 (35 days, $36{,}276{,}977$ samples) for training, 2024-02-17 ($1{,}004{,}815$) for validation, and 2024-02-18 ($963{,}523$) for testing. Each exposure constitutes one sample; the historical events associated with a request are never counted as samples. The original `effective_view` feedback is kept as the label and is not recomputed from watch time. The two numerical fields, video `duration` and `content_age_seconds`, are transformed by $\log(1+x)$ on their valid values and then standardized with training statistics; a non-positive duration or an invalid upload time is treated as missing, in which case the value is set to 0 and a missing indicator is retained. The 15 categorical fields cover user/device and demographic attributes, video/author/category attributes, and

time buckets, and their vocabularies are likewise fitted on the training split; for entity fields shared between the exposure and the history side, such as video and author ids, the training vocabulary is fitted from both the training-period exposures and their valid history.

Behavior sequences in RecFlow. Historical events are joined to exposures through the `request_id` of the same day. We retain only events that are strictly earlier than the timestamp of the current request and that do not belong to the current request itself, which prevents temporal leakage. From each historical event we extract five fields—video id, author id, the primary and secondary category, and upload type—and keep at most 50 steps. Sequences are left-padded so that the most recent behavior always occupies the rightmost position, and an accompanying mask marks the valid steps. The `request_id` is used only for joining histories and for GAUC grouping and is never fed to the model as a feature; the post-exposure feedback of the current request and the intermediate stage outputs of the original ranking system are excluded from the model inputs as well.