Hierarchical LoRA MoE for Efficient CTR Model Scaling

Zhichen Zeng[1] Mengyue Hang[2] Xiaolong Liu[2] Xiaoyi Liu[2] Xiao Lin[1] Ruizhong Qiu[1] Tianxin Wei[1]
Zhining Liu[1] Siyang Yuan[2] Chaofei Yang[2] Yiqun Liu[2] Hang Yin[2] Jiyan Yang[2] Hanghang Tong[1]
[1] University of Illinois Urbana-Champaign, [2] Meta AI
zhichenz@illinois.edu, hangm@meta.com

Abstract

Deep models have driven significant advances in click-through rate (CTR) prediction. While *vertical scaling* via layer stacking improves model expressiveness, the layer-by-layer sequential computation poses challenges to efficient scaling. Conversely, *horizontal scaling* through Mixture of Experts (MoE) achieves efficient scaling by activating a small subset of experts in parallel, but flat MoE layers may struggle to capture the hierarchical structure inherent in recommendation tasks. To push the Return-On-Investment (ROI) boundary, we explore the complementary strengths of both directions and propose HiLoMoE, a hierarchical LoRA MoE framework that enables holistic scaling in a parameter-efficient manner. Specifically, HiLoMoE employs lightweight rank-1 experts for parameter-efficient horizontal scaling, and stacks multiple MoE layers with hierarchical routing to enable combinatorially diverse expert compositions. Unlike conventional stacking, HiLoMoE routes based on prior layer scores rather than outputs, allowing all layers to execute in parallel. A principled three-stage training framework ensures stable optimization and expert diversity. Experiments on four public datasets show that HiLoMoE achieving better performance-efficiency trade-off, achieving an average AUC improvement of 0.20% in AUC and 18.5% reduction in FLOPs compared to the non-MoE baseline.

Keywords

Recommendation, Mixture of Experts, LoRA, Model Scaling

1 Introduction

Click-through rate (CTR) prediction serves as a foundational task in modern recommender systems, enabling platforms to rank items, allocate ad impressions, and personalize content delivery [33, 40, 46, 78**?**]. As the user and system scale continue to grow, there has been an increasing demand for more expressive and efficient CTR models that can make accurate predictions under tight latency and resource constraints [4, 67, 81, 83–86]. To meet this demand, *model scaling*, the ability to grow model capacity in a performance- and cost-effective manner, emerges as a critical challenge [36, 87–89].

In recent years, we have observed extensive efforts on *vertical scaling* via layer stacking [82, 87, 88], where deeper networks expand model capacity and allow for more complex representations. Despite noticeable performance gains, vertical scaling inherently requires sequential computation, leading to substantial inference latency as depth increases. In addition, stacking more layers often introduces a large number of parameters, making vertical scaling less suitable for resource-constrained scenarios where parameter efficiency is critical. Moreover, vertical scaling lacks conditional computation as every input passes through the full stack of layers, leaving little room for personalized computation that is crucial for CTR prediction where user interests and contexts vary widely.

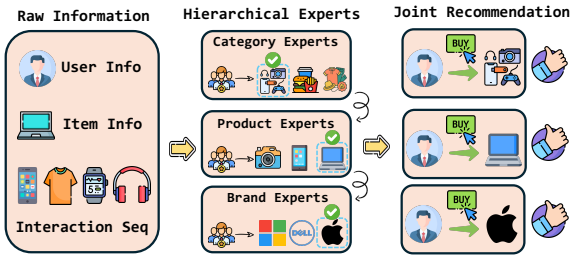


Figure 1: Motivation for Hierarchical MoE. Recommendation follows a hierarchical structure, where user–item interactions span multiple granularities. By hierarchically selecting experts such as `Electronics` (category), `Laptop` (product), and `Apple` (brand), the model collaboratively delivers personalized recommendations at different levels, effectively leveraging hierarchical knowledge for more accurate predictions.

To enable more efficient model scaling, recent works explore *horizontal scaling* based on the mixture of experts (MoE) framework, [12, 101] and have achieved great success in large language models [23, 43]. MoE introduces multiple experts and dynamically selects a *small subset* for each input, enabling personalized computation while significantly reducing inference cost and memory usage. However, in CTR prediction, both user and item information often exhibit a strong hierarchical structure that flat, single-layer MoE may fail to capture [94]. As shown in Figure 1, item features typically include high-level category information (e.g., electronics or apparel), followed by mid-level product information (e.g., camera or laptop), and finally finer-grained brand identity (e.g., Apple or Microsoft). Such multi-level semantics are difficult to disentangle within a single-layer MoE, often leading to under-specialized experts and limited expressiveness. This motivates the need for *hierarchically-structured expert designs* that can better model the compositional nature of inputs and promote expert diversity.

To address these challenges, it is essential to explore complementary roles of vertical and horizontal scaling. In this paper, we propose HiLoMoE, a hierarchical LoRA MoE framework that enables *holistic model scaling* in both vertical and horizontal directions in a parameter-efficient manner, which is built upon three key innovations. First (*LoRA experts*), to support efficient horizontal scaling, we represent each expert as a rank-1 perturbation of a shared base weight matrix, which significantly reduces parameter and memory overhead, allowing us to scale the number of experts without inflating the overall model size. Second (*hierarchical routing*), to achieve efficient vertical scaling, we introduce a hierarchical routing mechanism, where the expert selection at each layer conditions on the routing scores from previous layers. For one thing, such hierarchically-structured design enables combinatorially diverse expert paths, boosting model expressiveness without increasing

per-layer width. For another, the conditional routing computation depends only on routing scores rather than the intermediate expert outputs, enabling parallel inference across layers. Third (*training framework*), to train the complex system, we propose a three-stage training pipeline, augmented with auxiliary losses and controlled gradient flow, to stabilize training and promote expert diversity.

The main contributions of this paper are summarized as follows:

- **Model design.** We propose a hierarchical LoRA MoE framework to achieve efficient scaling for Transformer-based CTR models. The LoRA experts support efficient horizontal scaling for personalized computation, while the hierarchical routing mechanism enables vertical scaling through deep expert compositions with parallelizable inference.
- **Training framework.** We introduce a principled training framework, consisting of a three-stage training pipeline for stable optimization, and auxiliary losses to ensure balanced expert utilization and effective specialization.
- **Experiments.** Benchmark experiments suggest that HiLoMoE consistently enhances Transformer-based models by an average of 0.20% in AUC and 18.5% reduction in FLOPs compared to non-MoE counterparts. Besides, HiLoMoE exhibits promising scaling in depth and width, with performance increasing alongside the number of layers and experts.

2 Related Works

In the era of big data and AI [34, 37–39], improving recommendation quality while controlling model complexity is a core challenge [24, 48, 93], since even minor gains in prediction accuracy brings substantial improvements in revenue and user experience [68, 70, 72–75, 79]. In this section, we review related works on CTR prediction, MoE and LoRA as the foundation to understand HiLoMoE.

2.1 CTR Prediction Models

Click-Through Rate (CTR) prediction estimates the likelihood of a user clicking an item, an essential component in recommender systems. Early models [8, 15, 35, 57, 63, 76, 95] follow the factorization machine paradigm, combining memorization and generalization by fusing linear models with deep neural networks. With richer user sequences, Transformer-based models [40–42, 49, 56] become prominent for capturing sequential dependencies and complex interactions. Models like BST employ Transformer encoders for behavior sequence modeling [5], and DIN [97] and DIEN [96] employed attention mechanisms and evolving interest to dynamically focus on relevant user actions. Building on these foundations, TransAct [67] blends real-time user actions with long-term interests via a hybrid real-time/batch modeling framework, LiRank [4] employs ensembles of multiple interaction modules in large-scale ranking systems, CARL [7] accelerates computation through cache-aware mechanisms, END4Rec [17] introduces an efficient miner and denoising modules for multi-behavior sequences, SRP4CTR [16] bridges pre-trained sequential encoders with CTR models via cross-attention, and InterFormer [82] enhances cross-modal integration via interleaved information flow.

2.2 MoE and LoRA

To achieve efficient model scaling, mixture of experts and low-rank adaptation have served as prominent approaches.

MoE architecture increases model capacity by dynamically selecting a small subset of experts for each input, enabling personalized computation while maintaining inference efficiency. Early works [1, 22, 25, 52] utilize MoE to decompose the large task into smaller ones to enhance efficiency. Recently, MoE has attracted extensive attention due to its success in efficient scaling in large language models [11, 12, 23, 43]. Sparsely-gated MoE layers [12, 28, 55] achieve significant improvement in model capacity with minor efficiency loss. Auxiliary losses are further proposed to achieve stable model training [101] and balanced expert loading [12].

In parallel, LoRA [19, 42, 45, 90] enables efficient model adaptation by injecting low-rank parameter matrices into frozen networks, delivering full fine-tuning performance with far fewer trainable parameters. Modular strategies [20, 53, 58] utilize LoRA for multi-task learning through dynamic module fusion. To reduce redundancy and parameter overhead, different works adopt weight tying [54], random adaptation [2, 27], and federated-efficient designs [21].

Recently, MoE and LoRA have converged in hybrid designs to unlock richer adaptability and task efficiency. Early examples [14, 65] blend multiple adaptation modules in each Transformer layer using MoE-style routing to boost performance without additional inference. More advanced designs include task-aware routing [44, 47], asymmetric LoRA experts [60, 64], and composite expert architectures for better performance–efficiency tradeoffs [29, 66, 80].

2.3 MoE on CTR Models

Recent CTR models leverage MoE frameworks to enhance scalability and context sensitivity. Mixture of LoRA [13, 71, 77] embed domain-specific adapters as experts and route inputs dynamically. Multi-task and multi-domain modeling efforts [50, 92] decompose knowledge into shared, domain-specific, and task-specific experts, enabling flexible cross-domain personalized adaptation. Personalization and modality-aware routing are explore [3, 32, 51, 69, 100], mixing textual, behavioral, auxiliary, and modality-biased experts for diverse user contexts. Hierarchical MoE architectures [30, 59, 91] use multi-level gating structures to sequentially integrate shared, task-specific, and temporal information. To encourage expert diversity, D-MoE [62] introduces a cross-expert de-correlation loss.

3 Preliminaries

We use bold uppercase letters for matrices (e.g., $\boldsymbol{X}$), bold lowercase letters for vectors (e.g., $\boldsymbol{x}$), and lowercase letters for scalars (e.g., n). We use superscripts to denote layer index, e.g., $\boldsymbol{u}^{(l)}$ and subscripts to denote expert index, e.g., $\boldsymbol{u}_i$. User set and item set are denoted by $\mathcal{U}$ and $\mathcal{I}$, respectively. The interaction sequence of a user $u \in \mathcal{U}$ is denoted as $S_u = [i_{u,1}, \ldots, i_{u,T}] \in \mathcal{S}$, where $i_{u,t} \in \mathcal{I}$. We use $y_{i_{u,t}} \in \{0, 1\}$ to indicate whether the user u clicked on the item $i_{u,t}$ at timestamp t.

CTR prediction. CTR prediction estimates the probability of a user clicking on an item given heterogeneous information, e.g., sparse categorical features, numerical features, and interaction sequences. The goal of CTR prediction is to learn a function $f\colon \mathcal{U} \times \mathcal{I} \times \mathcal{S} \to [0, 1]$ such that $\Pr(y_{i_{u,T+1}} = 1 \mid u, i_{u,T+1}, S_u; \theta) = f(u, i_{u,T+1}, S_u; \theta)$ with high accuracy.

Mixture of Experts. An MoE layer consists of K experts $\{E_k\}_{k=1}^{K}$ and a gating function $g\colon \mathbb{R}^d \to \Delta^{K-1}$ that outputs probabilities over

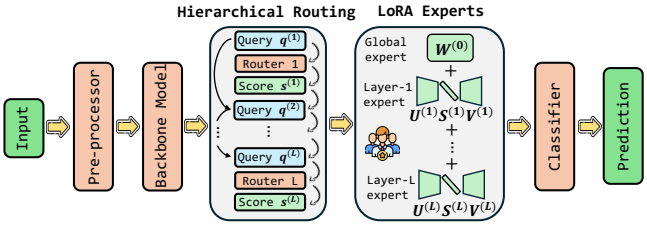


Figure 2: An overview of the proposed HiLoMoE which includes L layers, each containing K rank-1 experts.

experts. Given an input $h \in \mathbb{R}^d$, the output is a weighted sum of expert outputs:

$$\mathrm{MoE}(h) = \sum_{k=1}^{K} g_k(h) E_k(h).$$

Low-Rank Adaptation. LoRA introduces a pair of low-rank matrices $(\boldsymbol{A}, \boldsymbol{B})$ into a linear layer $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ by parameterizing

$$\boldsymbol{W}' = \boldsymbol{W} + \boldsymbol{B}\boldsymbol{A},$$

where $\boldsymbol{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\boldsymbol{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ with rank $r \ll \min(d_{\text{in}}, d_{\text{out}})$. In general, $\boldsymbol{W}$ is the pre-trained model parameter, which remains frozen during fine-tuning, and $\boldsymbol{A}, \boldsymbol{B}$ are learnable parameters for fine-tuning. This reduces trainable parameters to $O(r(d_{\text{in}} + d_{\text{out}}))$ and allows efficient storage of multiple task adapters.

4 Methodology

In this section, we introduce our proposed HiLoMoE, a hierarchical LoRA mixture of experts architecture for holistic model scaling. We first provide an overview of the model architecture in Section 4.1. Afterwards, we introduce the parameter-efficient LoRA expert design in Section 4.2, followed by the hierarchical routing mechanism in Section 4.3. A principled training framework is further introduced in Section 4.4.

4.1 Model Overview

Figure 2 presents an overview of the proposed HiLoMoE framework, which is designed to achieve parameter-efficient and scalable modeling through a combination of *LoRA experts* and a *hierarchical routing mechanism*.

First, to enable efficient horizontal scaling, HiLoMoE employs LoRA experts, where each expert is implemented as a rank-1 low-rank approximation of a full-rank weight matrix. This reduces the parameter complexity of each expert from quadratic to linear with respect to hidden size, making the model lightweight yet expressive.

Second, to facilitate efficient vertical scaling, HiLoMoE introduces a hierarchical routing mechanism that enables expert selection at each layer based on both the input query and the routing decisions from previous layers. Unlike conventional layer stacking where each layer is computed sequentially, HiLoMoE decouples lightweight routing score computation from computationally heavy expert execution. Specifically, we compute the lightweight routing scores layer-by-layer in a hierarchical fashion, but execute the heavy expert transformation computations *in parallel* across all layers, leading to significant efficiency gains during inference.

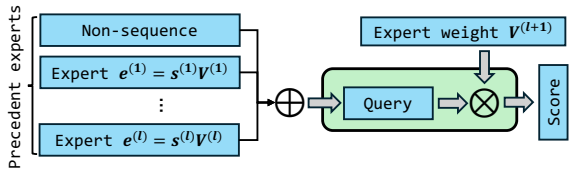


Figure 3: A hierarchical routing strategy selects the current experts based on selections from the previous layers.

Together, these two design choices make HiLoMoE a compelling solution for scalable CTR prediction models that balance expressiveness, efficiency, and modularity.

4.2 Parameter-Efficient LoRA experts

We first introduce our parameter-efficient LoRA experts. LoRA has been widely adopted in multi-domain settings, where a shared base weight serves as the foundation for capturing generalizable knowledge, while lightweight low-rank adapters specialize in domain-specific nuances. The low-rank formulation not only improves efficiency by reusing the heavy base weights across different adapters, but also enhances robustness: the shared backbone provides stable, broadly applicable representations, while the low-rank updates enable fine-grained adaptation to each domain.

Given that CTR models often operate under tight FLOPs and latency budgets, rank-1 experts are adopted to achieve extreme parameter efficiency for high-frequency inference. Specifically, for the l-th layer, each expert is the product of two vectors $\boldsymbol{W}_i^{(l)} = \boldsymbol{u}_i^{(l)} \boldsymbol{v}_i^{(l)\mathsf{T}}, \forall i = 1, 2, ..., K$, where $\boldsymbol{u}_i^{(l)}, \boldsymbol{v}_i^{(l)} \in \mathbb{R}^d$. Given a routing score $\boldsymbol{s}^{(l)} \in \Delta^{K-1}$, the l-th layer expert weight $\boldsymbol{W}^{(l)}$ is defined as

$$\boldsymbol{W}^{(l)} = \sum_{i=1}^{K} \boldsymbol{s}_i^{(l)} \boldsymbol{W}_i^{(l)} = \boldsymbol{U}^{(l)} \mathrm{diag}\left(\boldsymbol{s}^{(l)}\right) \boldsymbol{V}^{(l)},$$

where $\boldsymbol{U}^{(l)} = \left[\boldsymbol{u}_1^{(l)} \| \cdots \| \boldsymbol{u}_K^{(l)}\right] \in \mathbb{R}^{d \times K}, \boldsymbol{V}^{(l)} = \left[\boldsymbol{v}_1^{(l)} \| \cdots \| \boldsymbol{v}_K^{(l)}\right]^{\mathsf{T}} \in \mathbb{R}^{K \times d}$ are the concatenations of expert weights, and $\boldsymbol{W}^{(0)}$ is the shared expert weight. And the final transformation weight is the addition of the expert weights at each layer and a shared base weight $\boldsymbol{W}^{(0)}$, that is

$$\boldsymbol{W} = \boldsymbol{W}^{(0)} + \sum_{l=1}^{L} \boldsymbol{U}^{(l)} \mathrm{diag}\left(\boldsymbol{s}^{(l)}\right) \boldsymbol{V}^{(l)}. \tag{1}$$

During inference, the expert weights $\boldsymbol{U}^{(l)}, \boldsymbol{V}^{(l)}$ at each layer remain fixed, and the final transformation weight is determined solely by the layer-wise routing scores $\boldsymbol{s}^{(l)}$. As we will detail in the next section, the computation of routing scores is decoupled from the outputs of the MoE layers and instead depends only on the routing scores from preceding layers. This decoupling enables us to pre-compute all routing scores in a lightweight forward pass. Subsequently, the expert contributions across all layers can be aggregated and the sequence transformation applied in a single operation. As a result, *vertical scaling incurs no additional inference cost*, since the heavy expert computations across layers can be executed in parallel after routing is resolved.

4.3 Hierarchical Routing Strategy

Unlike conventional MoE frameworks that primarily scale in width by increasing the number of experts, HiLoMoE introduces a new paradigm that enables simultaneous depth–width scaling. This design enables the model to benefit from deep hierarchical transformations (depth) and combinatorially diverse expert selections (width), enhancing expressiveness and scalability.

To this end, our router design is guided by the following three principles. First (*hierarchical routing*), The routing decision at each layer is conditioned on the routing outcomes from preceding layers, forming a hierarchical selection process. This structure encourages the model to extract information in a coarse-to-fine manner. For example, in CTR prediction tasks, earlier layers may focus on high-level attributes such as item categories, while later layers refine these decisions based on fine-grained features like item brand or user intent. Second (*parallel inference*), to ensure scalability without incurring additional inference cost, different MoE layers are designed to execute in parallel. Third (*heterogeneous interaction*), CTR models often receive heterogeneous information, e.g., sequence and non-sequence information, and effective interaction among these features is at the core of the success of CTR prediction [82].

To meet these principles, we introduce the hierarchical routing design shown in Figure 3. Specifically, non-sequence information is processed by a query projection matrix $\boldsymbol{W}_{\text{proj}}$ to extract a low-dimensional representation $\boldsymbol{x} \in \mathbb{R}^d$ as the initial query $\boldsymbol{q}^{(1)}$.

For l-th layer router, it takes the input query $\boldsymbol{q}^{(l)}$ as input and calculates the routing score by the inner product of input query and expert weights $\boldsymbol{V}^{(l)}$, that is

$$\boldsymbol{s}^{(l)} = \text{Softmax}\left(\boldsymbol{q}^{(l)}\boldsymbol{V}^{(l)}/\sqrt{d}\right).$$

To enable hierarchical routing, the key is to design a proper query $\boldsymbol{q}^{(l)}$ that incorporates expert information from previous layers. For each layer, we compute the expert representation $\boldsymbol{e}^{(l)}$ using the expert weights $\boldsymbol{V}^{(l)}$ averaged by the routing score $\boldsymbol{s}^{(l)}$, i.e., $\boldsymbol{e}^{(l)} = \boldsymbol{s}^{(l)}\boldsymbol{V}^{(l)}$. The input query $\boldsymbol{q}^{(l+1)}$ for next layer is further updated by the sum of previous query $\boldsymbol{q}^{(l+1)}$ and expert representation $\boldsymbol{e}^{(l)}$:

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + \boldsymbol{e}^{(l)} = \boldsymbol{x} + \sum_{i=1}^{l} \boldsymbol{e}^{(i)}. \tag{2}$$

Such query update and routing score computation bring two notable advantages. For one thing, hierarchical routing enables structured, layer-dependent expert selection, which encourages combinatorially diverse expert compositions across layers and enhances model expressiveness. For another, the routing score computation is decoupled from the actual expert transformation, allowing the full set of selected expert transformations to be executed simultaneously after routing scores are determined, which substantially reduces the computation overhead typically associated with vertical scaling.

4.4 Principled Training Framework

Apart from model design, a principled training framework is essential to enable stable and effective model scaling. We address this challenge from two aspects, including a progressive warmup strategy to stabilize training, and auxiliary losses to promote balanced and diverse expert utilization.

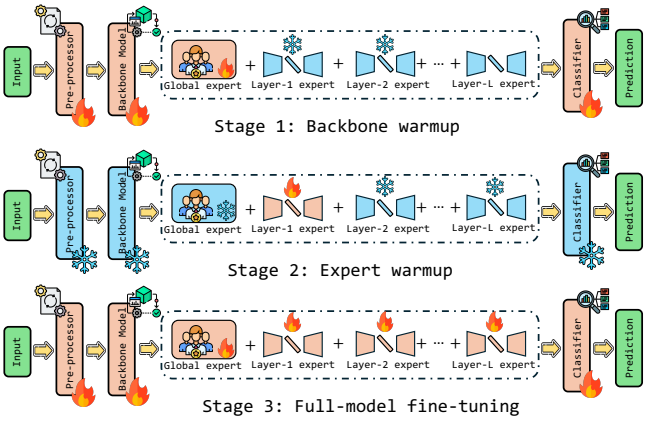


Figure 4: A 3-stage training framework for stable training.

Model Warmup. Directly training the full model can be unstable due to the intricate dependencies across layers and experts To mitigate this, we adopt a 3-stage training pipeline shown in Figure 4 to progressively build the model in a bottom-up fashion.

In the first stage (*backbone warmup*), we train only the backbone components,including the pre-processor, backbone model, classifier, and the shared expert. This stage is equivalent to training a *non-MoE baseline model* and serves as a stable initialization for the subsequent stages.

In the second stage (*expert warmup*), we sequentially activate and train the LoRA experts layer-by-layer, while freezing the backbone and previously activated experts. To avoid interference from untrained experts, experts are initialized with zero-valued LoRA weights, i.e., $\boldsymbol{V}^{(l)} \sim \mathcal{N}(0, I), \boldsymbol{U}^{(l)} \leftarrow \boldsymbol{0}$. This ensures that each layer is trained as an *incremental refinement* over earlier representations, leading to more stable optimization and faster convergence.

In the final stage (*full-model fine-tuning*), we fine-tune the entire model jointly, updating all parameters from the well-initialized starting point provided by the previous stages. This final step ensures global consistency and maximizes model performance.

Auxiliary Losses. Another critical challenge is to ensure balanced expert utilization. Without appropriate load-balancing mechanisms, the router may consistently favor a small subset of experts, while leaving others underutilized. This phenomenon, known as expert collapse [9, 12], undermines the intended modularity of MoE by effectively reducing it to a single-expert non-MoE architecture, thereby limiting its capacity and scalability.

To address this issue, we adopt two auxiliary losses, including load-balancing loss ℓ_{lb} [12] and z-loss ℓ_z [101], defined as follows

$$\begin{aligned} &\text{load-balancing loss} : \ell_{lb} = \frac{N}{B \cdot A} \sum_{b,k} \boldsymbol{s}(b,k) \cdot \mathbb{1}_{b,k}, \\ &\text{z-loss} : \ell_z = \text{logsumexp}(\boldsymbol{s}), \end{aligned} \tag{3}$$

where B is the batch size, A is the number of activated experts, and $\mathbb{1}_{b,k}$ is the indicator function denoting whether expert k is activated for sample b. The load-balancing loss encourages a uniform distribution of routing scores by penalizing over-reliance on a subset of experts, promoting balanced expert utilization. The z-loss regularizes the scale of gating logits, preventing excessively large values that lead to numerical instability and degraded training dynamics.

Although the auxiliary losses in Eq. (3) are designed to encourage diverse and balanced expert utilization, improper use may lead to conflicts with the main prediction objective, ultimately degrading model performance. To mitigate this risk, we isolate the influence of the auxiliary losses by *restricting their gradient flow*. Specifically, we allow gradients from ℓ_{lb} and ℓ_z to update only the router parameters, while prohibiting their influence on other model components such as the backbone network and expert weights. This targeted gradient routing ensures that auxiliary losses focus solely on improving routing quality, without interfering with the optimization of the primary prediction task.

4.5 Analysis

To better understand the benefits of HiLoMoE, we first provide analysis on model complexity as follows.

Proposition 4.1 (Complexity Analysis). *For an L-layer K-expert HiLoMoE operating on a d-dimensional sequence of length N, the space complexity is $O(KLd)$ and the time complexity is $O(Nd^2)$.*

For space complexity, HiLoMoE scales linearly with respect to the number of experts K, the number of layers L and the feature dimension d. This represents a significant improvement over conventional MoE frameworks [12, 28, 101], which typically incur quadratic parameter costs in d, as well as over LoRA-based MoEs [60, 66], which introduce an additional scaling factor proportional to the LoRA rank r. For time complexity, HiLoMoE scales linearly with the sequence length N and quadratically with the feature dimension d, matching the computational profile of standard Transformer models. Importantly, due to the parallel design of HiLoMoE, vertical scaling via additional MoE layers does not increase the time complexity. This is because the computationally expensive expert transformations are executed in parallel and fused into a single transformation step after routing scores are precomputed. In contrast, conventional vertical scaling requires sequential layer-wise computation, resulting in time complexity that grows linearly with the number of layers L. Thus, HiLoMoE enables expressive deep scaling without incurring additional inference cost.

Besides, we analyze how the number of expert combinations scale as the the number of experts per layer K, the number of HiLoMoE layers L, and the number of activated experts A increase.

Proposition 4.2 (Model Scaling). *For an L-layer K-expert HiLoMoE with Top-A expert activation, the number of possible expert combinations is $\binom{K}{A}^L$. Specifically:*

- *Layer scaling: When K, A are fixed with $1 \le A < K$, expert complexity grows exponentially w.r.t. L.*
- *Expert scaling: When A, L are fixed, expert complexity is $\Theta(K^{AL})$, which is polynomial w.r.t. K.*
- *Activation scaling: When K, L are fixed, expert complexity is $O\left(\left(\frac{eK}{A}\right)^{AL}\right)$, which is sublinear w.r.t. A.*

Proposition 4.2 reveals the expressive power of HiLoMoE, demonstrating that even with modest expert counts and sparse activation, hierarchical routing can yield exponentially large and diverse expert paths, which is crucial for balancing efficiency and capacity.

5 Experiments

We carry out extensive experiments to answer the following research questions:

- **RQ1:** How effective is the proposed HiLoMoE on benchmark CTR datasets? (Section 5.2)
- **RQ2:** How does model performance scale with HiLoMoE in horinzontal and vertical directions? (Section 5.3)
- **RQ3:** To what extent does HiLoMoE improve the performance-efficiency tradeoff? (Section 5.4)
- **RQ4:** To what extent does HiLoMoE benefit from different model and training framework designs? (Section 5.5)

5.1 Experiment Setup

Dataset. We evaluate on three public benchmark datasets, including AmazonElectronics[18], TaobaoAd[61], KuaiVideo[31], and MicroVideo [6], which are summarized in Appendix C.

Baseline Methods. We consider 5 state-of-the-art Transformer-based CTR models as the backbone model, including DIN [97], DIEN [96], BST [5], TransAct [67] and InterFormer [82]. Besides, we compare HiLoMoE with 4 MoE and LoRA baselines including Non-MoE, Switch Transformer [12], MoLE [66] and HydraLoRA [60].

Metrics. We adopt three metrics to evaluate the model performance and efficiency, including:

- `AUC` provides an aggregated measure of model capacity in correctly classifying positive and negative samples across all thresholds. Higher the better.
- `LogLoss` (cross-entropy loss) measures the distance between the prediction $\hat{y}$ and the label y, and is computed as $L(y, \hat{y}) = -(y \log(\hat{y}) + (1-y) \log(1-\hat{y}))$. Lower the better.
- `#Params` provides the number of parameters in the MoE module.

5.2 Benchmark Results

To evaluate the effectiveness of HiLoMoE, we compare its performance against state-of-the-art MoE methods on various Transformer-based CTR models across multiple benchmark datasets. For fair comparison, we adopt consistent model configurations on different MoE methods. We report the results in Table 1, from which we draw the following observations:

(1) HiLoMoE achieves state-of-the-art performance across all models and datasets. Compared to the non-MoE baseline, HiLoMoE attains the highest AUC in 16 out of 20 cases and the second highest in three additional cases. For LogLoss, it ranks best in 13 cases and second-best in four. These results show that HiLoMoE consistently enhances prediction performance with minimal additional complexity, validating its robustness on diverse settings.

(2) Compared to the non-MoE baseline, HiLoMoE achieves consistent gains in prediction quality while maintaining compact model size. Across all CTR models and datasets, HiLoMoE achieves an average improvement of 0.15% on AUC and 0.13% on LogLoss. This performance improvement is achieved with only a modest increase of 6.17K parameters on average, which includes the low-rank expert weights and router components. These findings highlight HiLoMoE's ability to provide meaningful performance improvements with limited overhead, making it well-suited for deployment in resource-constrained environments.

Table 1: Benchmark results. We report AUC (↑), LogLoss (↓), and the parameter count of the MoE FFN layer (↓). Blue and red cells indicate the best and second-best performance, respectively (we exclude Non-MoE baseline for #Params comparison as it is inherently the most parameter-efficient). For fair comparison, all non-MoE and MoE variants adopt the same configuration.

Dataset		TaobaoAd			AmazonElectronics			KuaiVideo			MicroVideo		
	Model + MoE	AUC	LogLoss	#Params	AUC	LogLoss	#Params	AUC	LogLoss	#Params	AUC	LogLoss	#Params
BST	Non-MoE	0.6486	0.1937	7.35K	0.8683	0.4571	24.83K	0.7457	0.4353	10.58K	0.7253	0.4171	2.13K
	Switch	0.6501	0.1935	35.61K	0.8671	0.4632	107.78K	0.7458	0.4347	46.46K	0.7260	0.4168	12.67K
	MoLE	0.6494	0.1937	26.40K	0.8662	0.4729	58.63K	0.7462	0.4341	48.00K	0.7255	0.4171	14.21K
	HydraLoRA	0.6484	0.1938	16.15K	0.8668	0.4622	42.24K	0.7451	0.4346	26.50K	0.7238	0.4191	9.09K
	HiLoMoE	0.6505	0.1932	11.90K	0.8699	0.4562	35.39K	0.7463	0.4343	17.71K	0.7254	0.4162	6.96K
DIN	Non-MoE	0.6468	0.1931	6.27K	0.8762	0.4487	2.11K	0.7431	0.4386	4.13K	0.7262	0.4156	8.32K
	Switch	0.6475	0.1929	31.49K	0.8780	0.4427	10.63K	0.7431	0.4392	20.68K	0.7267	0.4149	41.73K
	MoLE	0.6478	0.1929	23.67K	0.8760	0.4440	10.59K	0.7433	0.4374	22.22K	0.7263	0.4150	29.31K
	HydraLoRA	0.6472	0.1930	15.71K	0.8795	0.4408	5.95K	0.7432	0.4367	10.92K	0.7266	0.4155	20.06K
	HiLoMoE	0.6475	0.1928	13.35K	0.8789	0.4423	4.68K	0.7434	0.4380	9.16K	0.7271	0.4152	17.54K
DIEN	Non-MoE	0.6508	0.1930	6.27K	0.8803	0.4395	1.09K	0.7440	0.4366	4.13K	0.7220	0.4204	2.08K
	Switch	0.6508	0.1926	31.49K	0.8799	0.4457	5.51K	0.7444	0.4344	20.68K	0.7233	0.4193	10.44K
	MoLE	0.6511	0.1925	23.67K	0.8793	0.4421	7.00K	0.7446	0.4351	22.22K	0.7234	0.4190	11.98K
	HydraLoRA	0.6510	0.1927	15.71K	0.8804	0.4393	3.64K	0.7436	0.4374	10.92K	0.7231	0.4185	5.80K
	HiLoMoE	0.6513	0.1925	13.35K	0.8806	0.4390	2.54K	0.7446	0.4341	9.16K	0.7235	0.4192	4.68K
TransAct	Non-MoE	0.6488	0.1933	24.83K	0.8842	0.4366	33.09K	0.7446	0.4348	33.31K	0.7266	0.4151	16.67K
	Switch	0.6486	0.1927	105.73K	0.8828	0.4393	140.80K	0.7472	0.4343	141.57K	0.7265	0.4157	70.91K
	MoLE	0.6485	0.1929	56.58K	0.8824	0.4377	73.22K	0.7469	0.4333	95.49K	0.7254	0.4165	49.41K
	HydraLoRA	0.6491	0.1927	40.19K	0.8851	0.4340	52.74K	0.7470	0.4349	60.67K	0.7280	0.4137	30.98K
	HiLoMoE	0.6495	0.1926	33.34K	0.8860	0.4351	44.22K	0.7473	0.4331	46.43K	0.7281	0.4130	23.39K
InterFormer	Non-MoE	0.6515	0.1926	1.07K	0.8829	0.4297	2.11K	0.7421	0.4369	2.13K	0.7165	0.4173	2.13K
	Switch	0.6513	0.1926	6.40K	0.8854	0.4299	12.67K	0.7451	0.4351	12.67K	0.7167	0.4169	12.67K
	MoLE	0.6523	0.1925	7.94K	0.8827	0.4325	12.68K	0.7430	0.4361	14.21K	0.7167	0.4176	14.21K
	HydraLoRA	0.6521	0.1925	4.86K	0.8845	0.4296	8.58K	0.7434	0.4366	9.09K	0.7169	0.4166	9.09K
	HiLoMoE	0.6541	0.1924	3.57K	0.8874	0.4279	6.82K	0.7450	0.4359	6.96K	0.7178	0.4161	6.96K

(3) Relative to existing state-of-the-art MoE baselines (Switch Transformer [12], MoLE [66], HydraLoRA [60]), HiLoMoE consistently shows superior efficiency and effectiveness. On average, it improves AUC by 0.08% and reduces LogLoss by 0.10% compared to the best competing MoE variant (HydraLoRA). At the same time, HiLoMoE reduces parameter count by an average of 4.04K, which is equivalent to a 21.0% reduction relative to the most parameter-efficient MoE competitor (HydraLoRA). The improved accuracy can be attributed to HiLoMoE's combinatorially diverse expert compositions enabled by hierarchical routing, while the efficiency gains stem from its low-rank expert formulation. These results show that HiLoMoE offers a much more efficient MoE configuration with on par or even enhanced accuracy.

5.3 Model Scaling

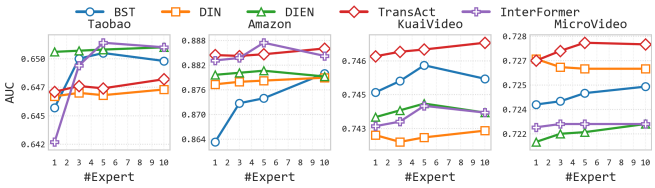
Model scaling, improving performance as capacity grows with minimal saturation, is a key property for CTR models. We evaluate HiLoMoE under horizontal (#experts) and vertical (#layers) scaling.

5.3.1 Horizontal Scaling. We first evaluate the horizontal scaling by varying the number of experts in {1,3,5,10}. As shown in Figure 5, **increasing experts generally improves AUC and reduces LogLoss, confirming the effectiveness of horizontal scaling.**

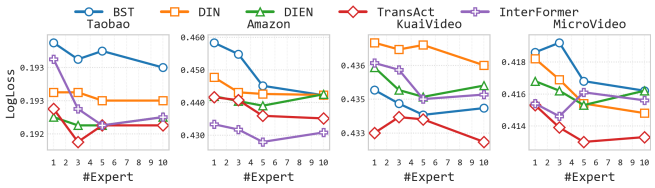
Across the majority of models and datasets, we observe a monotonic or near-monotonic gain in AUC as the number of experts increases. Similarly, we observe a steady decline in LogLoss as the number of experts increases, indicating not only improved ranking performance but also better-calibrated probability estimates. This trend shows enhanced model expressiveness brought by expert diversity, leading to more accurate prediction. For example, increasing the number of experts within a moderate range (e.g., 1 to 5) results in a clear AUC lift and LogLoss decline. When more experts are involved, the performance either continues to improve or plateaus afterward, suggesting that a small number of experts is already sufficient to bring meaningful gains, especially when paired with efficient routing and training strategies. In addition, we observe no significant degradation when more experts are added, demonstrating the robustness against overparameterization.

5.3.2 Vertical Scaling. We also evaluate the vertical scaling of HiLoMoE by varying the number of MoE layers in {1,2,3}. The results in Figure 6 suggest that **modest vertical scaling leads to incremental improvements in AUC and LogLoss, though the gains are generally smaller compared to horizontal scaling.**

Across most models and datasets, increasing the number of MoE layers from one to two consistently leads to noticeable performance gains, confirming that deeper expert composition can enhance representational capacity. While gains may taper or fluctuate slightly beyond two layers, this behavior reflects the need for architectural balance, neither too deep nor too shallow, rather than an inherent

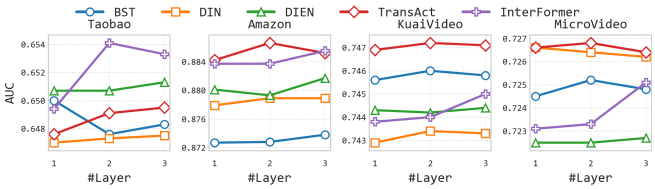


(a) Horizontal scaling on AUC.

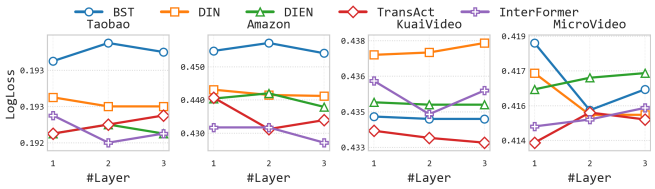


(b) Horizontal scaling on LogLoss.

Figure 5: Horizontal scaling on #experts.



(a) Vertical scaling on AUC.



(b) Vertical scaling on LogLoss.

Figure 6: Vertical scaling on #layers.

limitation of depth. Overall, these results suggest that vertical scaling is a promising and complementary direction within the broader HiLoMoE framework, enabling the construction of more expressive and efficient CTR models when combined with expert-level scaling.

Among the datasets, MicroVideo exhibits clearer vertical scaling benefits, especially in models like InterFormer and TransAct. This may be attributed to its long behavioral sequences and relatively compact feature space, which allow deeper transformations to be effectively stacked without overfitting. In contrast, on datasets such as TaobaoAd and KuaiVideo, the benefits of adding additional layers are marginal, possibly due to their shorter sequences or simpler patterns, which may not require deep hierarchical modeling.

Remark. We observe that vertical scaling yields limited or even negative gains compared to horizontal scaling. This is because stacking more HiLoMoE layers increases dependency depth and routing uncertainty, leading to redundant transformations and optimization instability. In contrast, horizontal scaling expands expert diversity at a fixed depth, aligning better with the heterogeneous yet shallow nature of user–item interactions in CTR tasks.

5.4 Performance-Efficiency Tradeoff

We evaluate the performance-efficiency tradeoff of HiLoMoE by comparing it with the non-MoE baseline. In general, we compute the relative improvements in AUC against both computational cost (FLOPs) and model size (#Params), using the non-MoE baseline as a reference point. We adopt the most state-of-the-art InterFormer model as the baseline, and fine-tune the non-MoE InterFormer configuration to achieve the best AUC. The non-MoE AUC, #params and FLOPs are further used as the reference points to compute the relative improvements, as shown in Figure 7.

(1) AUC vs FLOPs (left): HiLoMoE consistently achieves better performance and efficiency compared to the non-MoE baseline. Notably, on datasets like Taobao and MicroVideo, HiLoMoE reduces FLOPs by over 20% while still improving AUC by approximately 0.2%, highlighting its strong scalability and practical value in resource-constrained environments. Even on Amazon, where the reduction in FLOPs is minimal, we still observe a clear

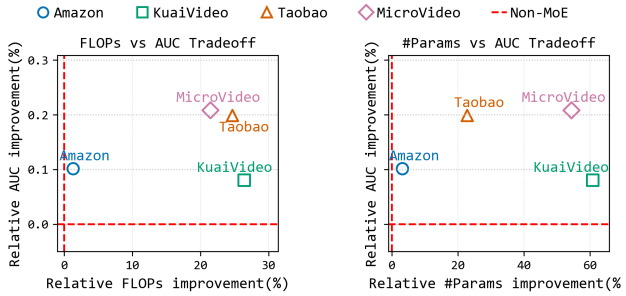


Figure 7: Performance-Efficiency Tradeoff. We report the relative improvements in AUC and FLOPs achieved by HiLoMoE on the InterFormer model, compared to its non-MoE counterpart. To ensure a fair comparison under optimal conditions, the baseline non-MoE model is scaled up to a larger configuration to reach the best AUC.

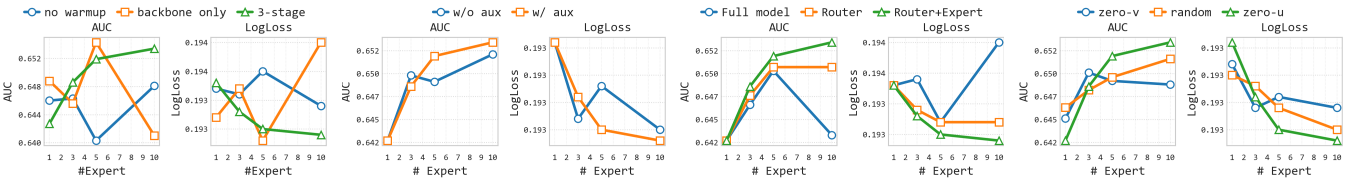
AUC gain, suggesting that HiLoMoE can deliver performance benefits even without sacrificing efficiency.

(2) AUC vs #Params (right): HiLoMoE exhibits substantial reductions in parameter count across all datasets, while achieving consistent AUC improvements. On KuaiVideo, HiLoMoE achieves a 60% reduction in #Params compared to the non-MoE baseline. This demonstrates the effectiveness of the LoRA-based expert design in enabling parameter-efficient modeling without compromising accuracy.

Overall, the tradeoff figures illustrate that HiLoMoE offers a highly favorable balance between accuracy and efficiency. By leveraging low-rank expert parameterization and sparse hierarchical routing, HiLoMoE successfully reduces resource consumption while enhancing model performance, making it well-suited for large-scale CTR prediction systems in both training and deployment scenarios.

5.5 Studies

In this section, we carry out studies on model warmup (Section 5.5.1), auxiliary losses (Section 5.5.2) and model initialization (Section 5.5.3).



(a) Study on model warmup. **(b) Study on auxiliary losses.** **(c) Study on aux loss gradients.** **(d) Study on model initialization.**

Figure 8: Studies on HiLoMoE. We compare model performance with (a) different training strategy, (b) auxiliary losses, (c) auxiliary loss gradient flows, and (d) model initialization. In general, HiLoMoE trained under 3-stage training framework, augmented with auxiliary losses and initialized with *zero-u* initialization yields the best performance.

5.5.1 On the effect of model warmup. We first study the effect of model warmup, and results are shown in Figure 8a. We consider three variants: (1) *no warmup*, where the whole model is trained directly without warmup, (2) *backbone only*, where only the backbone model is warmed up while the experts are not, and (3) *3-stage*, where we perform the three-stage training illustrated in Figure 4.

When the numbers of experts increases, *no warmup* leads to unstable performance, with AUC fluctuating and LogLoss remaining relatively high. *Backbone only* provides partial improvement, but its scaling performance is less stable with performance degrading as the number of experts grows. In contrast, the proposed *3-stage* warmup achieves the most stable scaling behavior, as AUC increases steadily, while LogLoss decreases smoothly. These results highlight that progressive warmup not only stabilizes optimization but also allows the experts to be more effectively integrated with the backbone, thereby unlocking the benefits of expert scaling.

5.5.2 On the effect of auxiliary losses. We then compare model performance with and without auxiliary losses, and results are shown in Figure 8b. Overall, incorporating auxiliary losses consistently improves both AUC and LogLoss. Without auxiliary losses, the model exhibits less stable scaling as the number of experts increases, with AUC and LogLoss fluctuating when the number of experts increases from 3 to 10. By contrast, applying auxiliary losses yields a smoother scaling curve: AUC increases steadily, while LogLoss decreases monotonically. This suggests that auxiliary objectives effectively regularize the routing mechanism, preventing expert collapse and promoting more balanced utilization. As a result, the experts contribute more complementary knowledge, leading to measurable gains in both predictive accuracy and calibration.

Besides, we study how different gradient flows of auxiliary losses affect model performance. We consider three different variants, including (1) *full model*, where auxiliary losses affect the training of the whole model, (2) *router*, where the auxiliary losses only affect the router update, and (3) *router+expert*, where the auxiliary losses affect both router and expert update. We observe that letting the auxiliary losses propagate through the *full model* leads to unstable behavior: while model performance peaks with 5 experts, it quickly drops when the number of experts increases. Restricting the auxiliary losses to update only the *router* yields a more stable trend, with AUC reaching 0.650 and LogLoss stabilizing around 0.193, but further gains are limited as expert scaling increases. In contrast, applying auxiliary losses to both the *router+expert* consistently delivers the best performance: AUC steadily improves beyond 0.652 at 10 experts, while LogLoss decreases smoothly to below 0.193. These results confirm that auxiliary objectives are most effective when guiding both routing and expert specialization, while constraining their gradient flow away from the backbone avoids interference with the main prediction task.

5.5.3 On the effect of expert initialization. Finally, we study the effect of different expert initialization strategies, and results are shown in Figure 8d. We consider three variants: (1) *zero-v*, where $\boldsymbol{V}^{(l)}$ is zero-initialized and $\boldsymbol{U}^{(l)}$ is randomly initialized, (2) *random*, where both $\boldsymbol{V}^{(l)}$ and $\boldsymbol{U}^{(l)}$ are randomly initialized, and (3) *zero-u*, where $\boldsymbol{U}^{(l)}$ is zero-initialized and $\boldsymbol{V}^{(l)}$ is randomly initialized.

Among the three variants, *zero-v* leads to the weakest performance: AUC quickly saturates and LogLoss remains relatively high. This degradation is mainly because $\boldsymbol{V}^{(l)}$ is used to compute the average expert representation and input queries for hierarchical routing in Eq. (2), and zero-initializing $\boldsymbol{V}^{(l)}$ thus diminishes expert diversity and limits expressiveness in the early training stage. In contrast, *random* initialization provides a stronger baseline, with AUC gradually improving and LogLoss decreasing smoothly as the number of experts increases. The best performance is achieved by *zero-u*, where the input direction $\boldsymbol{U}^{(l)}$ is zero-initialized while $\boldsymbol{V}^{(l)}$ is randomly initialized: AUC surpasses 0.652 at 10 experts, and LogLoss consistently decreases to below 0.193. These results suggest that freezing the input direction at initialization suppresses noisy expert activations in early training, while allowing the output direction to adapt flexibly, thereby stabilizing optimization and improving convergence.

6 Conclusion

In this paper, we present HiLoMoE, a hierarchical LoRA MoE framework designed to achieve holistic model scaling for Transformer-based CTR prediction. By combining low-rank expert parameterization with hierarchical routing, HiLoMoE supports efficient horizontal and vertical scaling while maintaining parameter efficiency and parallelizable inference. To ensure stable optimization, we introduce a three-stage training framework with auxiliary losses to encourage expert diversity and balanced utilization. Extensive experiments across multiple benchmark datasets and backbone architectures demonstrate that HiLoMoE consistently improves predictive accuracy while reducing computational cost, achieving an average of 0.20% gains in AUC and an average 18.5% reduction in FLOPs compared to non-MoE baselines. These results highlight the promise of structured and parameter-efficient MoE designs for advancing CTR prediction under practical efficiency constraints.

References

[1] Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. 2025. Resmoe: Space-efficient compression of mixture of experts llms via residual restoration. *arXiv preprint arXiv:2503.06881* (2025).

[2] Daniel Bershatsky, Daria Cherniuk, Talgat Daulbaev, Aleksandr Mikhalev, and Ivan Oseledets. 2024. LoTR: Low tensor rank weight adaptation. *arXiv preprint arXiv:2402.01376* (2024).

[3] Shuqing Bian, Xingyu Pan, Wayne Xin Zhao, Jinpeng Wang, Chuyuan Wang, and Ji-Rong Wen. 2023. Multi-modal mixture of experts represetation learning for sequential recommendation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 110–119.

[4] Fedor Borisyuk, Mingzhou Zhou, Qingquan Song, Siyu Zhu, Birjodh Tiwana, Ganesh Parameswaran, Siddharth Dangi, Lars Hertel, Qiang Xiao, Xiaochen Hou, et al. 2024. LiRank: Industrial Large Scale Ranking Models at LinkedIn. *arXiv preprint arXiv:2402.06859* (2024).

[5] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data*. 1–4.

[6] Xusong Chen, Dong Liu, Zheng-Jun Zha, Wengang Zhou, Zhiwei Xiong, and Yan Li. 2018. Temporal hierarchical attention at category-and item-level for micro-video click-through prediction. In *Proceedings of the 26th ACM international conference on Multimedia*. 1146–1153.

[7] Xiaoshuang Chen, Gengrui Zhang, Yao Wang, Yulin Wu, Shuo Su, Kaiqiao Zhan, and Ben Wang. 2024. Cache-Aware Reinforcement Learning in Large-Scale Recommender Systems. In *Companion Proceedings of the ACM on Web Conference 2024*. 284–291.

[8] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[9] Zewen Chi, Li Dong, Shaohan Huang, Damai Dai, Shuming Ma, Barun Patra, Saksham Singhal, Payal Bajaj, Xia Song, Xian-Ling Mao, et al. 2022. On the representation collapse of sparse mixture of experts. *Advances in Neural Information Processing Systems* 35 (2022), 34600–34613.

[10] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

[11] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International conference on machine learning*. PMLR, 5547–5569.

[12] William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research* 23, 120 (2022), 1–39.

[13] Dehong Gao, Shufan Chen, Zhiming Yang, Luwei Yang, Haining Gao, Muyang Wu, Shanqing Yu, Qi Xuan, Wenxiao Zhang, Libin Yang, et al. 2025. MLoRA+: Transformer-fusion Mixture-of-LoRA Network for Multi-domain Click-Through Rate Prediction. *Expert Systems with Applications* (2025), 129486.

[14] Yunhao Gou, Zhili Liu, Kai Chen, Lanqing Hong, Hang Xu, Aoxue Li, Dit-Yan Yeung, James T Kwok, and Yu Zhang. 2023. Mixture of cluster-conditional lora experts for vision-language instruction tuning. *arXiv preprint arXiv:2312.12379* (2023).

[15] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. *arXiv preprint arXiv:1703.04247* (2017).

[16] Ruidong Han, Qianzhong Li, He Jiang, Rui Li, Yurou Zhao, Xiang Li, and Wei Lin. 2024. Enhancing CTR Prediction through Sequential Recommendation Pre-training: Introducing the SRP4CTR Framework. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 3777–3781.

[17] Yongqiang Han, Hao Wang, Kefan Wang, Likang Wu, Zhi Li, Wei Guo, Yong Liu, Defu Lian, and Enhong Chen. 2024. End4rec: Efficient noise-decoupling for multi-behavior sequential recommendation. *arXiv preprint arXiv:2403.17603* (2024).

[18] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv preprint arXiv:2106.09685* (2021).

[20] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269* (2023).

[21] Nam Hyeon-Woo, Moon Ye-Bin, and Tae-Hyun Oh. 2021. Fedpara: Low-rank hadamard product for communication-efficient federated learning. *arXiv preprint arXiv:2108.06098* (2021).

[22] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[23] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[24] Yue Jiang, Haokun Lin, Yang Bai, Bo Peng, Zhili Liu, Yueming Lyu, Yong Yang, Jing Dong, et al. 2025. Image-level Memorization Detection via Inversion-based Inference Perturbation. In *The Thirteenth International Conference on Learning Representations*.

[25] Michael I Jordan and Robert A Jacobs. 1994. Hierarchical mixtures of experts and the EM algorithm. *Neural computation* 6, 2 (1994), 181–214.

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

[27] Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2023. Vera: Vector-based random matrix adaptation. *arXiv preprint arXiv:2310.11454* (2023).

[28] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668* (2020).

[29] Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, et al. 2024. Mixlora: Enhancing large language models fine-tuning with lora-based mixture of experts. *arXiv preprint arXiv:2404.15159* (2024).

[30] Danwei Li, Zhengyu Zhang, Siyang Yuan, Mingze Gao, Weilin Zhang, Chaofei Yang, Xi Liu, and Jiyan Yang. 2023. Adatt: Adaptive task-to-task fusion network for multitask learning in recommendations. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4370–4379.

[31] Yongqi Li, Meng Liu, Jianhua Yin, Chaoran Cui, Xin-Shun Xu, and Liqiang Nie. 2019. Routing micro-videos via a temporal graph-guided recommendation system. In *Proceedings of the 27th ACM international conference on multimedia*. 1464–1472.

[32] Yuchen Li, Hao Zhang, Yongqi Zhang, Hengyi Cai, Mingxin Cai, Shuaiqiang Wang, Haoyi Xiong, Linghe Kong, Dawei Yin, and Lei Chen. 2025. RankExpert: A Mixture of Textual-and-Behavioral Experts for Multi-Objective Learning-to-Rank in Web Search. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4578–4589.

[33] Zihao Li, Dongqi Fu, Mengting Ai, and Jingrui He. 2025. APEX^2: Adaptive and Extreme Summarization for Personalized Knowledge Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins (Eds.). ACM, 741–752. doi:10.1145/3690624.3709213

[34] Zihao Li, Zhichen Zeng, Xiao Lin, Feihao Fang, Yanru Qu, Zhe Xu, Zhining Liu, Xuying Ning, Tianxin Wei, Ge Liu, et al. 2025. Flow Matching Meets Biology and Life Science: A Survey. *arXiv preprint arXiv:2507.17731* (2025).

[35] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. 2018. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1754–1763.

[36] Mingfu Liang, Xi Liu, Rong Jin, Boyang Liu, Qiuling Suo, Qinghai Zhou, Song Zhou, Laming Chen, Hua Zheng, Zhiyuan Li, et al. 2025. External large foundation model: How to efficiently serve trillions of parameters for online ads recommendation. In *Companion Proceedings of the ACM on Web Conference 2025*. 344–353.

[37] Haokun Lin, Teng Wang, Yixiao Ge, Yuying Ge, Zhichao Lu, Ying Wei, Qingfu Zhang, Zhenan Sun, and Ying Shan. 2025. Toklip: Marry visual tokens to clip for multimodal comprehension and generation. *arXiv preprint arXiv:2505.05422* (2025).

[38] Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems* 37 (2024), 87766–87800.

[39] Haokun Lin, Haobo Xu, Yichen Wu, Ziyu Guo, Renrui Zhang, Zhichao Lu, Ying Wei, Qingfu Zhang, and Zhenan Sun. 2025. Quantization meets dllms: A systematic study of post-training quantization for diffusion llms. *arXiv preprint arXiv:2508.14896* (2025).

[40] Jianghao Lin, Bo Chen, Hangyu Wang, Yunjia Xi, Yanru Qu, Xinyi Dai, Kangning Zhang, Ruiming Tang, Yong Yu, and Weinan Zhang. 2024. Clickprompt: CTR models are strong prompt generators for adapting language models to CTR prediction. In *Proceedings of the ACM Web Conference 2024*. 3319–3330.

[41] Xiao Lin, Zhining Liu, Dongqi Fu, Ruizhong Qiu, and Hanghang Tong. 2024. Backtime: Backdoor attacks on multivariate time series forecasting. *Advances in Neural Information Processing Systems* 37 (2024), 131344–131368.

[42] Xiao Lin, Zhichen Zeng, Tianxin Wei, Zhining Liu, Hanghang Tong, et al. 2025. Cats: Mitigating correlation shift for multivariate time series classification. *arXiv preprint arXiv:2504.04283* (2025).

[43] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024.

Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).
[44] Qidong Liu, Xian Wu, Xiangyu Zhao, Yuanshao Zhu, Derong Xu, Feng Tian, and Yefeng Zheng. 2023. Moelora: An moe-based parameter efficient fine-tuning method for multi-task medical applications. *CoRR* (2023).
[45] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*.
[46] Xiaolong Liu, Zhichen Zeng, Xiaoyi Liu, Siyang Yuan, Weinan Song, Mengyue Hang, Yiqun Liu, Chaofei Yang, Donghyun Kim, Wen-Yen Chen, et al. 2024. A collaborative ensemble framework for ctr prediction. *arXiv preprint arXiv:2411.13700* (2024).
[47] Zefang Liu and Jiahua Luo. 2024. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. *arXiv preprint arXiv:2405.00361* (2024).
[48] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. 2023. Class-imbalanced graph learning without class rebalancing. *arXiv preprint arXiv:2308.14181* (2023).
[49] Ze Lyu, Yu Dong, Chengfu Huo, and Weijun Ren. 2020. Deep match to rank model for personalized click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 156–163.
[50] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.
[51] Van-Khang Nguyen, Duc-Hoang Pham, Huy-Son Nguyen, Cam-Van Thi Nguyen, Hoang-Quynh Le, and Duc-Trong Le. 2025. Multi-modal Adaptive Mixture of Experts for Cold-start Recommendation. *arXiv preprint arXiv:2508.08042* (2025).
[52] Steven Nowlan and Geoffrey E Hinton. 1990. Evaluation of adaptive mixtures of competing experts. *Advances in neural information processing systems* 3 (1990).
[53] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning. *arXiv preprint arXiv:2005.00247* (2020).
[54] Adithya Renduchintala, Tugrul Konuk, and Oleksii Kuchaiev. 2023. Tied-lora: Enhancing parameter efficiency of lora with weight tying. *arXiv preprint arXiv:2311.09578* (2023).
[55] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey Hinton, and Jeffrey Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arXiv preprint arXiv:1701.06538* (2017).
[56] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*. 1441–1450.
[57] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the web conference 2021*. 2828–2837.
[58] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5227–5237.
[59] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*. 269–278.
[60] Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems* 37 (2024), 9565–9584.
[61] Tianchi. [n. d.]. Ad display/click data on taobao.com, 2018. https://tianchi.aliyun.com/dataset/56.
[62] Jiancheng Wang, Mingjia Yin, Junwei Pan, Ximei Wang, Hao Wang, and Enhong Chen. 2025. Enhancing CTR Prediction with De-correlated Expert Networks. *arXiv preprint arXiv:2505.17925* (2025).
[63] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*. 1785–1797.
[64] Xujia Wang, Haiyan Zhao, Shuo Wang, Hanqing Wang, and Zhiyuan Liu. 2024. Malora: Mixture of asymmetric low-rank adaptation for enhanced multi-task learning. *arXiv preprint arXiv:2410.22782* (2024).
[65] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models. *arXiv preprint arXiv:2205.12410* 1, 2 (2022), 4.
[66] Xun Wu, Shaohan Huang, and Furu Wei. 2024. Mixture of lora experts. *arXiv preprint arXiv:2404.13628* (2024).
[67] Xue Xia, Pong Eksombatchai, Nikil Pancha, Dhruvil Deven Badani, Po-Wei Wang, Neng Gu, Saurabh Vishwas Joshi, Nazanin Farahpour, Zhiyuan Zhang, and Andrew Zhai. 2023. Transact: Transformer-based realtime user action model for recommendation at pinterest. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5249–5259.
[68] Haobo Xu, Yuchen Yan, Dingsu Wang, Zhe Xu, Zhichen Zeng, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. 2024. Slog: An inductive spectral graph neural network beyond polynomial filter. In *Forty-first International Conference on Machine Learning*.
[69] Jiahui Xu, Lu Sun, and Dengji Zhao. 2024. MoME: Mixture-of-masked-experts for efficient multi-task recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2527–2531.
[70] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. 2024. Discrete-state continuous-time diffusion for graph generation. *Advances in Neural Information Processing Systems* 37 (2024), 79704–79740.
[71] Ken Yagel, Eyal German, and Aviel Ben Siman Tov. 2025. MoE-MLoRA for Multi-Domain CTR Prediction: Efficient Adaptation with Expert Specialization. *arXiv preprint arXiv:2506.07563* (2025).
[72] Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Xiaoting Li, Zhe Xu, Zhichen Zeng, Lihui Liu, Zhining Liu, and Hanghang Tong. 2024. Thegcn: Temporal heterophilic graph convolutional network. *arXiv preprint arXiv:2412.16435* (2024).
[73] Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM Web Conference 2024*. 2485–2496.
[74] Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2023. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems* 36 (2023), 6844–6861.
[75] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. Bright: A bridging algorithm for network alignment. In *Proceedings of the web conference 2021*. 3907–3917.
[76] Carl Yang, Lanxiao Bai, Chao Zhang, Quan Yuan, and Jiawei Han. 2017. Bridging collaborative filtering and semi-supervised learning: a neural approach for poi recommendation. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 1245–1254.
[77] Tianxiao Yang et al. 2024. MLoRA: Multi-domain Low-Rank Adaptation for Click-through Rate Prediction. In *Proceedings of the Web Conference*.
[78] Hyunsik Yoo, Zhichen Zeng, Jian Kang, Ruizhong Qiu, David Zhou, Zhining Liu, Fei Wang, Charlie Xu, Eunice Chan, and Hanghang Tong. 2024. Ensuring user-side fairness in dynamic recommender systems. In *Proceedings of the ACM Web Conference 2024*. 3667–3678.
[79] Qi Yu, Zhichen Zeng, Yuchen Yan, Lei Ying, R Srikant, and Hanghang Tong. 2025. Joint optimal transport and embedding for network alignment. In *Proceedings of the ACM on Web Conference 2025*. 2064–2075.
[80] Hanqing Zeng, Yinglong Xia, Zhuokai Zhao, Gilbert Jiang, Qiang Zhang, Jiayi Liu, Lizhu Zhang, Xiangjun Fan, and Benyu Zhang. 2025. S'MoRE: Structural Mixture of Residual Experts for LLM Fine-tuning. *arXiv preprint arXiv:2504.06426* (2025).
[81] Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. 2024. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 16660–16668.
[82] Zhichen Zeng, Xiaolong Liu, Mengyue Hang, Xiaoyi Liu, Qinghai Zhou, Chaofei Yang, Yiqun Liu, Yichen Ruan, Laming Chen, Yuxin Chen, et al. 2024. Interformer: Towards effective heterogeneous interaction learning for click-through rate prediction. *arXiv preprint arXiv:2411.09852* (2024).
[83] Zhichen Zeng, Ruizhong Qiu, Wenxuan Bao, Tianxin Wei, Xiao Lin, Yuchen Yan, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. 2025. Pave Your Own Path: Graph Gradual Domain Adaptation on Fused Gromov-Wasserstein Geodesics. *arXiv preprint arXiv:2505.12709* (2025).
[84] Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. 2024. Graph mixup on approximate gromov–wasserstein geodesics. In *Forty-first International Conference on Machine Learning*.
[85] Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. 2023. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM web conference 2023*. 372–382.
[86] Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. 2023. Generative graph dictionary learning. In *International Conference on Machine Learning*. PMLR, 40749–40769.
[87] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. 2024. Wukong: Towards a Scaling Law for Large-Scale Recommendation. *arXiv preprint arXiv:2403.02545* (2024).
[88] Buyun Zhang, Liang Luo, Xi Liu, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, et al. 2022. DHEN: A deep and hierarchical ensemble network for large-scale click-through rate prediction. *arXiv preprint arXiv:2203.11014* (2022).

[89] Guoxiao Zhang, Yi Wei, Yadong Zhang, Huajian Feng, and Qiang Liu. 2025. Balancing Efficiency and Effectiveness: An LLM-Infused Approach for Optimized CTR Prediction. In *Companion Proceedings of the ACM on Web Conference 2025*. 596–600.

[90] Jinghan Zhang, Junteng Liu, Junxian He, et al. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems* 36 (2023), 12589–12610.

[91] Shengzhe Zhang, Liyi Chen, Dazhong Shen, Chao Wang, and Hui Xiong. 2025. Hierarchical Time-Aware Mixture of Experts for Multi-Modal Sequential Recommendation. In *Proceedings of the ACM on Web Conference 2025*. 3672–3682.

[92] Zijian Zhang, Shuchang Liu, Jiaao Yu, Qingpeng Cai, Xiangyu Zhao, Chunxu Zhang, Ziru Liu, Qidong Liu, Hongwei Zhao, Lantao Hu, et al. 2024. M3oe: Multi-domain multi-task mixture-of experts recommendation framework. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 893–902.

[93] Lecheng Zheng, Baoyu Jing, Zihao Li, Zhichen Zeng, Tianxin Wei, Mengting Ai, Xinrui He, Lihui Liu, Dongqi Fu, Jiaxuan You, et al. 2024. Pyg-ssl: A graph self-supervised learning toolkit. *arXiv preprint arXiv:2412.21151* (2024).

[94] Zuowu Zheng, Changwang Zhang, Xiaofeng Gao, and Guihai Chen. 2022. HIEN: hierarchical intention embedding network for click-through rate prediction. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 322–331.

[95] Guorui Zhou, Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Na Mou, Xinchen Luo, et al. 2020. CAN: revisiting feature co-action for click-through rate prediction. *arXiv preprint arXiv:2011.05625* (2020).

[96] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 5941–5948.

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

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

[99] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2020. Fuxictr: An open benchmark for click-through rate prediction. *CoRR* (2020).

[100] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 1121–1130.

[101] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906* (2022).

Appendix

A Proof

A.1 Proof of Proposition 4.2

Proposition 4.2 (Model Scaling). *For an L-layer K-expert HiLoMoE with Top-A expert activation, the number of possible expert combinations is $\binom{K}{A}^L$. Specifically:*

- *Layer scaling: When K, A are fixed with $1 \le A < K$, expert complexity grows exponentially w.r.t. L.*
- *Expert scaling: When A, L are fixed, expert complexity is $\Theta(K^{AL})$, which is polynomial w.r.t. K.*
- *Activation scaling: When K, L are fixed, expert complexity is $O\left(\left(\frac{eK}{A}\right)^{AL}\right)$, which is sublinear w.r.t. A.*

Proof. Each HiLoMoE layer with Top-A expert activation generates $\binom{K}{A}$ possible combinations, and stacking L layers further enlarges the number of combinations to $\binom{K}{A}^L$.

When K, A are fixed, it is obvious to show that the expert complexity grows exponential w.r.t. L.

Besides, we can bound $\binom{K}{A}$ as follows [10]:

$$\left(\frac{K}{A}\right)^A \le \binom{K}{A} \le \left(\frac{eK}{A}\right)^A$$

Therefore, when A, L are fixed, the expert complexity can be expressed by $N = \Theta(K^{AL})$, which is polynomial w.r.t. K.

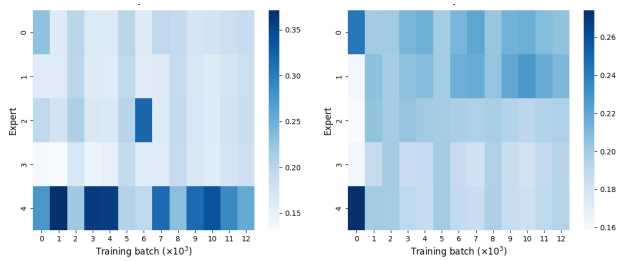
When K, L are fixed, according to the Stirling approximation [10]

$$\ln \binom{K}{A} = A \ln \frac{eK}{A} + O(\ln A),$$

which results in an expert complexity of $O\left(\frac{eK}{A}\right)^{AL}$ that is sublinear w.r.t. A. □

B Additional Experiments

We visualize the routing score along the training process with and without auxiliary losses in Figure 9. We adopt Transact as the backbone model, and average the routing score in each 1,000 batches. Darker color indicates the expert is more frequently activated.



(a) Routing score w/o aux losses. (b) Routing score w/ aux losses.

Figure 9: Routing score along the training process with and without auxiliary losses.

Without auxiliary losses (Figure 9a), the router exhibits clear mode collapse: load concentrates on a single expert for long stretches, sharply reducing expert diversity and, in the extreme, degenerating to a non-MoE model. In contrast, adding auxiliary losses (Figure 9b) yields markedly more even expert utilization over time. This prevents collapse, promotes expert specialization, and better exploits MoE diversity for personalized recommendation.

C Experiment Pipeline

C.1 Reproducibility

We adopt the FuxiCTR library [99] and BARS evaluation framework [98] for benchmark experiments, and conduct all experiments on NVIDIA A100 GPU.

For model configuration, we adopt the Top-1 activation for MoE. We fine-tune the number of HiLoMoE layers in {1,2,3}, the number of experts in {1,3,5,10}, and the embedding dimensions in {32,64,128}.

For model training, an Adam [26] optimizer with a learning rate scheduler is adopted for model optimization, where the initial learning rate is tuned in {1e-1, 1e-2, 1e-3}. We use a batch size of 2048, and train up to 100 epochs with early stop. For model warmup, we warm up the backbone model for 10,000 batches, each HiLoMoE layer for 5,000 batches, and fine-tune the full model till early stop.

For benchmark datasets, we adopt the default processing and configurations in BARS [98].

C.2 Dataset

Dataset statistics are summarized in Table 2, and detailed descriptions are provided as follows:

Table 2: Dataset Statistics.

Dataset	#Samples	#Feat. (Seq/Non-Seq)	Seq Length
Amazon	3.0M	5 (2/3)	100
TaobaoAd	25.0M	20 (3/17)	50
KuaiVideo	13.7M	6 (2/4)	100
MicroVideo	12.7M	6 (2/4)	100

- **AmazonElectronics** [18] is a large-scale dataset containing product reviews and metadata collected from Amazon's electronics category. It includes 192,403 users, 63,001 items, 801 categories, and a total of 1,689,188 interaction samples. The dataset features both non-sequential features, e.g., user ID, item ID, and item category, and sequence features, e.g., interacted items and corresponding categories, truncated or padded to a fixed length of 100. Following standard benchmark configurations, the dataset is split into 2.60 million training samples and 0.38 million testing samples.
- **TaobaoAds** [61] is a large-scale dataset consisting of 26 million ad click-through records collected over 8 consecutive days on Taobao, randomly sampled from 1,140,000 users. The dataset contains both non-sequential and sequential features. Non-sequential features include item-related attributes such as ad ID, category, and price, as well as user-related attributes such as user ID, gender, and age. Sequence features capture users' historical interactions, including sequences of item brands, categories, and behavior types, each truncated or padded to a fixed length of 50. Following the public benchmark configuration, we use 22.0 million samples for training and 3.1 million samples for testing.

- **KuaiVideo** [31] is a large-scale video interaction dataset comprising 3,239,534 user-video interactions from 10,000 users. It includes both non-sequence and sequence features. The non-sequence features consist of user ID, video ID, and pre-extracted visual embeddings of the videos. The sequence features capture user behavior histories, such as click, like, and skip interactions, with a fixed sequence length of 100. Following the public benchmark split, the dataset is divided into 10.9 million training samples and 2.7 million testing samples.
- **MicroVideo** [6] is a large-scale dataset collected from a popular micro-video sharing platform, comprising 12,737,619 user–video interactions from 10,986 users and 1,704,880 unique micro-videos. Each micro-video is associated with visual features extracted from its cover image using the Inception-v3 model, and a manually assigned category from a predefined set of 512 mutually exclusive categories. Interaction records include user ID, video ID, and timestamp, and cover both positive (clicked) and negative (skipped) feedback. Following the public benchmark split, the dataset consists of 8.9 million training samples and 3.8 million testing samples.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009