BIGGER IS NOT ALWAYS BETTER: TOWARDS EFFECTIVE SCALING OF SEQUENTIAL RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling model size has become an increasingly important approach to improving sequential recommendation. However, prior studies suggest that simply increasing model size does not necessarily lead to consistent performance improvements. As recommenders scale, additional model capacity does not necessarily yield proportional performance gains, while optimization becomes progressively more challenging. Therefore, a central challenge is to improve scaling efficiency and sustain consistent performance gains as model size increases. To address these, we propose TriScaleRec, a scaling-oriented framework for sequential recommendation. Specifically, TriScaleRec improves depth utilization through adaptive cross-depth representation aggregation. On this basis, it expands model capacity with sparse latent-space expert computation while controlling active computation, and further employs structure-aware optimization to sustain effective training at larger scales. Experiments on multiple public benchmarks show that TriScaleRec achieves strong recommendation performance while scaling experiments over six progressively larger model configurations further show that its performance continues to improve more consistently as model size increases.

1 INTRODUCTION

Sequential recommendation aims to predict users' future interactions from historical behaviors and has become a fundamental component of modern recommender systems (Quadrana et al., 2018; Wang et al., 2019). Over the years, the field has evolved from transition-based and recurrent modeling to convolutional and attention-based architectures, progressively improving the ability to capture temporal dependencies in user behavior (Rendle et al., 2010; Hidasi et al., 2016; Tang & Wang, 2018; Li et al., 2017). Building on these advances, Transformer-based methods such as SASRec (Kang & McAuley, 2018), BERT4Rec (Sun et al., 2019), and TiSASRec (Li et al., 2020) have substantially advanced sequential recommendation, while recent architectures such as HSTU (Zhai et al., 2024) further demonstrate the potential of increasing model capacity for large-scale recommendation.

However, simply making a model deeper or wider with more parameters does not necessarily lead to proportional performance gains in sequential recommendation (Zhang et al., 2024; Zivic et al., 2024). Recent studies further reveal substantial redundancy in large sequential recommendation models (Xu et al., 2025), suggesting that effective scaling depends on how additional model capacity is utilized rather than merely increased. This naturally raises a key question:

How can sequential recommenders scale more effectively as model size increases?

Effective model scaling exposes a sequence of increasingly important bottlenecks. As illustrated in Figure 1, conventional adjacent-layer propagation can progressively weaken shallow behavioral information as network depth increases, creating a gap between ideal and observed scaling behavior in increasingly deep models. Once depth can be more effectively utilized, further scaling shifts the bottleneck toward the feed-forward transformations, where increasing model capacity also increases active computation. Sparse Mixture-of-Experts methods such as Switch Transformer (Fedus et al., 2021) and LatentMoE (Elango et al., 2026) suggest that increasing capacity while controlling active computation is a promising direction. As the model becomes deeper and more parameterized, optimization further becomes more challenging at larger model scales because different parameter groups exhibit increasingly heterogeneous optimization dynamics.

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.

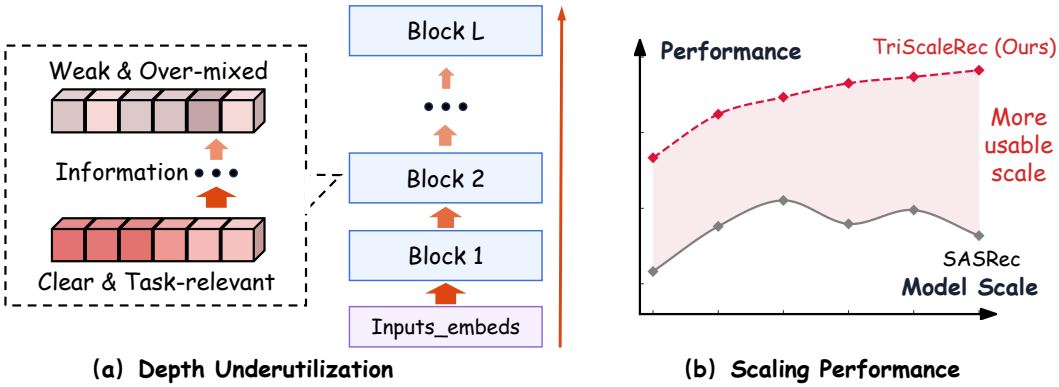


Figure 1: Motivation for effective scaling in sequential recommendation. (a) Repeated block-wise propagation weakens and over-mixes shallow behavioral information. (b) TriScaleRec achieves more consistent gains than SASRec as model scale increases.

To address these progressively emerging bottlenecks, we propose **TriScaleRec**, a scaling-oriented sequential recommender. We first enhance cross-depth information reuse to improve the utilization of additional depth. On this basis, sparse expert computation in a compact latent space expands model capacity while limiting the growth of active computation. Finally, we introduce a structure-aware optimization strategy to better accommodate the heterogeneous optimization dynamics that arise at larger model scales. Together, these designs progressively improve the utilization of depth, capacity, and optimization as model scale increases across a broader range of model configurations.

Our main contributions are summarized as follows:

- We revisit sequential recommendation scaling from the perspective of scaling efficiency, identifying inefficient depth utilization, increasing capacity–compute coupling, and more difficult large-scale optimization as three major bottlenecks.
- We propose **TriScaleRec**, which improves scaling efficiency through cross-depth information reuse and sparse latent-space expert computation, while alleviating optimization difficulties with a structure-aware hybrid optimization strategy.
- Extensive performance, scaling, efficiency, and mechanism analyses show that TriScaleRec achieves strong recommendation quality across diverse benchmark settings while obtaining more consistent gains over progressively larger configurations as model scale increases.

2 Related Work

Sequential Recommendation. Sequential recommendation models users' evolving preferences from chronologically ordered interactions. Early approaches explored transition-, recurrent-, and convolution-based architectures (Rendle et al., 2010; Hidasi et al., 2016; Tang & Wang, 2018; Li et al., 2017), while Transformer-based methods such as SASRec (Kang & McAuley, 2018), BERT4Rec (Sun et al., 2019), and TiSASRec (Li et al., 2020) substantially improved sequential dependency modeling. Subsequent studies further investigated self-supervised learning (Zhou et al., 2020; Xie et al., 2022; Qiu et al., 2022), efficient sequence modeling (Fan et al., 2021; Zhou et al., 2022; Liu et al., 2024b;a), and large-scale or long-sequence recommendation (Zhai et al., 2024; Chen et al., 2026; Li et al., 2026). More recently, BlossomRec (Ma et al., 2026) explores block-level sparse attention for sequential recommendation. Despite these advances, most existing methods primarily focus on improving sequence modeling, whereas our work focuses on how model capacity can be utilized more effectively as the recommender scales.

Scaling Sequential Recommendation. Recent studies have begun to investigate scaling behavior in sequential recommendation (Zhang et al., 2024; Zivic et al., 2024), showing that increasing model size does not necessarily translate into proportional performance gains. Related research provides several complementary directions for improving scaling efficiency. Residual and cross-layer mechanisms (He et al., 2016; Huang et al., 2017; Bachlechner et al., 2021; Wang et al., 2024; Kimi Team,

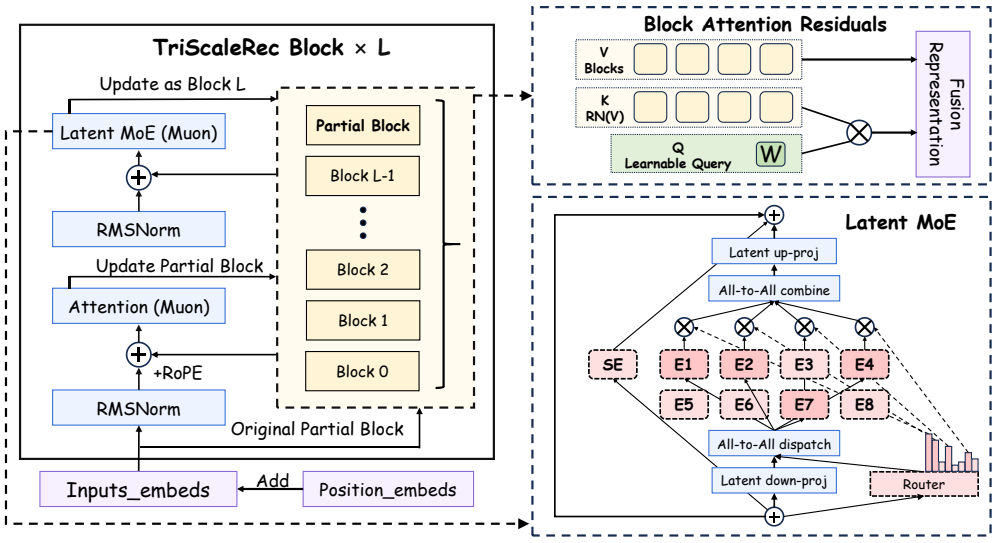


Figure 2: An overview of TriScaleRec architecture. TriScaleRec combines adaptive cross-depth representation aggregation, sparse latent-space expert computation, and hybrid structure-aware optimization to improve the utilization of depth, capacity, and optimization during model scaling.

2026; Cheng et al., 2026) improve information and gradient propagation in deep networks; sparse Mixture-of-Experts architectures (Shazeer et al., 2017; Lepikhin et al., 2020; Fedus et al., 2021; Dai et al., 2024; Elango et al., 2026) increase model capacity while controlling active computation; and optimization methods such as AdamW (Loshchilov & Hutter, 2019), Sophia (Liu et al., 2024c), and Muon (Jordan et al., 2024) explore more effective optimization for large models. Different from these works, TriScaleRec jointly addresses depth utilization, capacity–computation coupling, and optimization within a unified scaling-oriented framework for sequential recommendation.

3 METHOD

3.1 MODEL ARCHITECTURE OVERVIEW

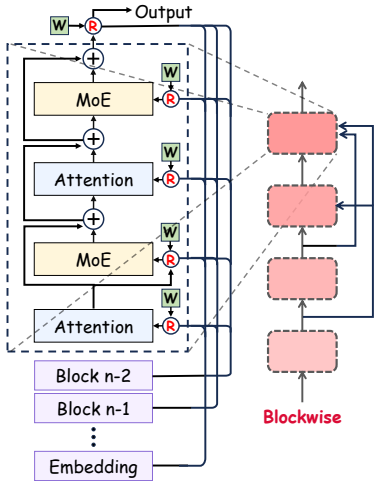


Figure 3: Adaptive Cross-Depth Aggregation.

As illustrated in Figure 2, TriScaleRec takes a user's historical interaction sequence as input and encodes it with a decoder-only Transformer (Vaswani et al., 2017) equipped with input-level positional embeddings and rotary positional embeddings (Su et al., 2024). While causal self-attention models sequential dependencies, relying only on adjacent-layer propagation can limit the effective use of deeper representations in increasingly deep networks. We therefore introduce adaptive cross-depth aggregation so that each layer can directly reuse informative representations from multiple preceding depths. As model capacity further increases, dense feed-forward transformations couple parameter growth with active computation, motivating sparse expert computation in a compact latent space (Elango et al., 2026) to expand capacity more efficiently. Larger models also become increasingly difficult to optimize, so we employ a hybrid optimization strategy that applies Muon (Jordan et al., 2024) to suitable dense matrix-valued parameters and AdamW (Loshchilov & Hutter, 2019) to the remaining parameters. The resulting hidden representation is finally used for next-item prediction.

3.2 BLOCK ATTENTION

Residual connections (He et al., 2016) alleviate optimization difficulties in deep networks, while recent Attention Residuals (Kimi Team, 2026) further demonstrate the potential of directly reusing representations across different depths. Conventional residual connections carry information forward but do not explicitly expose multiple earlier representations for selective reuse at each layer. In sequential recommendation, such access may help preserve shallow behavioral cues alongside deeper contextual patterns.

Inspired by this cross-depth information reuse principle, we employ **Block Attention**, which allows each computation stage to directly access and adaptively aggregate representations from multiple preceding depths. As illustrated in Figure 3, representations from previously completed blocks are directly exposed to the current computation stage and selectively fused according to the current sequence context, with position-specific weights over candidate depths.

To enable cross-depth information reuse without introducing excessive aggregation overhead, we assume the encoder contains L Transformer blocks, which are grouped into Fusion Blocks of size S. The b-th Fusion Block is defined as

$$\mathcal{B}_b = \{bS, bS+1, \ldots, (b+1)S-1\}. \tag{1}$$

Let F_b denote the output representation of the b-th completed Fusion Block, and $\widetilde{F}_{b,l}$ denote the intermediate representation at computation stage l of the current block. Block Attention constructs the depth representation set

$$\mathcal{D}_{b,l} = \{F_0, F_1, \ldots, F_{b-1}, \widetilde{F}_{b,l}\}. \tag{2}$$

To select informative representations from different depths, each computation stage maintains an independent learnable query $q_{b,l} \in \mathbb{R}^d$. For the j-th candidate representation $s_{t,j} \in \mathcal{D}_{b,l}$ at sequence position t, its relevance score is computed as

$$e^{b,l}_{t,j} = q^{\top}_{b,l}\,\mathrm{RMSNorm}(s_{t,j}), \qquad \alpha^{b,l}_{t} = \mathrm{Softmax}(e^{b,l}_{t}), \tag{3}$$

where $\alpha^{b,l}_{t,j}$ represents the contribution of the j-th depth representation. The updated representation is obtained by

$$\widetilde{F}_{b,l+1}(t) = \sum_j \alpha^{b,l}_{t,j} s_{t,j}. \tag{4}$$

Because the queries are layer-specific while the fusion weights depend on the current hidden representations, Block Attention can learn different depth preferences across computation stages and dynamically adapt them to different users and sequence positions. This provides direct cross-depth information paths and enables additional Transformer depth to be utilized more effectively.

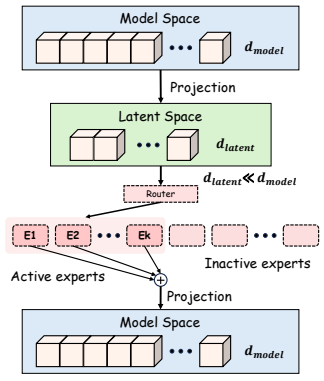


Figure 4: LatentMoE performs sparse expert computation in a compact latent space.

3.3 LATENTMOE

Sparse Mixture-of-Experts architectures (Shazeer et al., 2017; Fedus et al., 2021) alleviate this coupling by activating only a subset of experts for each input. Building on the **LatentMoE** design (Elango et al., 2026), we perform sparse expert computation in a compact latent space, decoupling expert capacity from the full model hidden dimension.

As illustrated in Figure 4, LatentMoE projects the input from the original d_{model}-dimensional space into a compact latent space of dimension d_{latent}, where $d_{\text{latent}} \ll d_{\text{model}}$. A router selects a small subset of experts for each input, and the activated experts transform the latent representation while the others remain inactive. Their outputs are aggregated and projected back to the original model space. By combining low-dimensional expert computation with sparse activation, LatentMoE expands model capacity while substantially limiting active computation.

Let $x_t \in \mathbb{R}^d$ denote the hidden representation at sequence position t, where d is the model dimension. To reduce the cost of expert computation before routing, LatentMoE first projects the representation into a compact latent space:

$$z_t = W_{\downarrow} x_t, \qquad W_{\downarrow} \in \mathbb{R}^{\ell \times d}, \tag{5}$$

where $\ell < d$ is the latent dimension. The resulting representation $z_t \in \mathbb{R}^{\ell}$ serves as the input to the routed experts. By performing expert transformations in the lower-dimensional latent space rather than directly in the d-dimensional model space, the parameter and activation cost of each expert is substantially reduced.

Given N routed experts, the router computes expert probabilities from the original hidden representation:

$$p_t = \mathrm{Softmax}\left(W_r x_t\right), \qquad W_r \in \mathbb{R}^{N \times d}. \tag{6}$$

Using the original representation for routing preserves the full hidden information when estimating expert relevance, while the subsequent expert computation remains confined to the compact latent space. This separates routing quality from expert computation cost.

Only the K experts with the highest routing scores are activated for each position. Denoting their indices by $\mathcal{T}_K(p_t)$, the latent expert output is

$$\hat{z}_t = \sum_{i \in \mathcal{T}_K(p_t)} p_{t,i} E_i(z_t), \tag{7}$$

where each expert $E_i(\cdot)$ operates entirely in the ℓ-dimensional latent space. Since only a small subset of experts is evaluated for each representation, increasing the total number of experts can expand model capacity without proportionally increasing active computation. The remaining experts are inactive for the current representation and introduce no expert computation.

The aggregated latent representation is then projected back to the original model space:

$$y_t = W_{\uparrow} \hat{z}_t, \qquad W_{\uparrow} \in \mathbb{R}^{d \times \ell}. \tag{8}$$

This shared up-projection restores the representation dimension required by the surrounding Transformer layers while keeping the routed transformations themselves lightweight.

Compared with conventional sparse MoE layers (Shazeer et al., 2017; Fedus et al., 2021), LatentMoE provides a more flexible capacity–computation trade-off. The latent dimension ℓ, total number of experts N, and active expert count K jointly control model capacity and active computation, enabling more compute-efficient scaling.

3.4 Hybrid Muon/AdamW Optimization

As model capacity increases, optimization becomes another factor that determines whether additional parameters can be effectively translated into recommendation quality. Different parameter groups in a sequential recommender, however, exhibit substantially different structures. In particular, the major transformations in self-attention and LatentMoE are dominated by dense matrix-valued parameters, whereas embeddings, normalization parameters, biases, and other auxiliary parameters have different geometries.

We therefore adopt a **Hybrid Muon/AdamW** strategy based on Muon (Jordan et al., 2024) and AdamW (Loshchilov & Hutter, 2019), rather than applying a single optimizer uniformly to the entire model.

Let Θ denote all trainable parameters. We partition them into two disjoint groups,

$$\Theta = \Theta_{\mathrm{Muon}} \cup \Theta_{\mathrm{AdamW}}, \qquad \Theta_{\mathrm{Muon}} \cap \Theta_{\mathrm{AdamW}} = \varnothing, \tag{9}$$

where Θ_{Muon} contains eligible dense matrix-valued parameters in the self-attention and LatentMoE transformations, and $\Theta_{\mathrm{AdamW}} = \Theta \setminus \Theta_{\mathrm{Muon}}$ contains the remaining parameters.

For a matrix parameter $W \in \Theta_{\mathrm{Muon}}$ with gradient G_t, Muon (Jordan et al., 2024) first accumulates a momentum update and applies a matrix-aware orthogonalization operator:

$$M_t = \mu M_{t-1} + G_t, \qquad \widehat{G}_t = \mathcal{O}(M_t), \tag{10}$$

where μ is the momentum coefficient and $\mathcal{O}(\cdot)$ denotes the orthogonalized update used by Muon. The parameter is then updated as

$$W_{t+1} = W_t - \eta_{\mathrm{M}} \widehat{G}_t, \tag{11}$$

with learning rate η_{M}. Parameters in Θ_{AdamW} are optimized using the standard AdamW update with learning rate η_{A} and decoupled weight decay.

This selective assignment concentrates Muon's matrix-aware updates on the two modules whose computation is primarily governed by large dense transformations, while retaining AdamW for parameter groups that do not naturally benefit from the same update geometry. As a result, the hybrid strategy aims to preserve the optimization benefit of Muon without introducing its update procedure indiscriminately across the entire model, providing a more targeted optimization scheme as TriScaleRec scales.

3.5 Prediction and Training Objective

Given the final hidden representation $h_t \in \mathbb{R}^d$ at sequence position t, we predict the next interacted item by measuring its compatibility with the corresponding item embedding. For a candidate item i with embedding e_i, the prediction score is computed as

$$s_{t,i} = h_t^\top e_i. \tag{12}$$

We optimize the model using the InfoNCE objective (van den Oord et al., 2018). Let i_t^+ denote the ground-truth next item and $\mathcal{N}_t$ denote the sampled negative set.

$$\mathcal{L} = -\sum_t \log \frac{\exp(s_{t,i_t^+}/\tau)}{\exp(s_{t,i_t^+}/\tau) + \sum_{j \in \mathcal{N}_t} \exp(s_{t,j}/\tau)}, \tag{13}$$

where τ is the temperature hyperparameter. The complete TriScaleRec model is trained end-to-end under this objective.

3.6 Theoretical Guarantees

We provide theoretical support for the two architectural components of TriScaleRec, with formal assumptions and proofs deferred to Appendix G.

Theorem 1 (Informal: Block Attention)**.** *Under the conditions specified in Appendix G, Block Attention approximates conventional adjacent-layer propagation with*

$$\|F_{\mathrm{BA}}(X) - F_{\mathrm{Adj}}(X)\|_F \le 2M\sqrt{n} \sum_{b=1}^{B-1} b e^{-\gamma_b} \prod_{r=b}^{B-1} L_r.$$

Thus, the approximation error vanishes as the attention margins increase. Moreover, under locally contractive adjacent propagation, cross-depth sensitivity may decay as $\mathcal{O}(\rho^{b-j})$, whereas Block Attention introduces a direct path whose sensitivity lower bound is independent of the depth distance $b - j$.

Theorem 2 (Informal: LatentMoE)**.** *If the active full-space experts admit an ℓ-dimensional shared latent approximation with residual $\epsilon_\ell(x)$, then*

$$\|F_{\mathrm{MoE}}(x) - F_{\mathrm{Latent}}(x)\|_2 \le \epsilon_\ell(x)\|x\|_2.$$

Meanwhile, its active expert computation scales as $\mathcal{O}(d\ell + Kr\ell^2)$, compared with $\mathcal{O}(Krd^2)$ for full-space experts.

These results characterize how Block Attention improves cross-depth information preservation and how LatentMoE decouples expert capacity from active computation.

4 Experiments

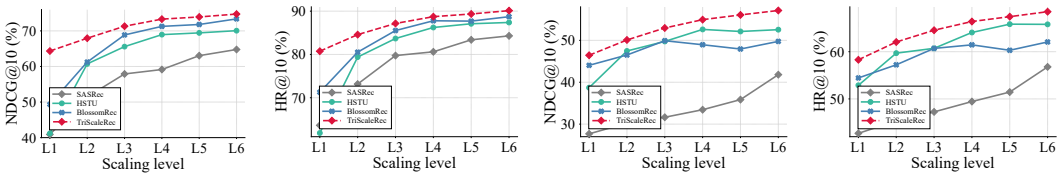
4.1 Experimental Setup

Datasets. We evaluate TriScaleRec on four public sequential recommendation benchmarks spanning diverse scales and interaction patterns: ML-1M (Harper & Konstan, 2015), Amazon (He &

Table 1: Performance comparison of TriScaleRec with representative sequential recommendation methods. Best results are highlighted in bold. Gray cells denote TriScaleRec results, while darker red cells indicate larger relative improvements over the average baseline performance.

Model	ML-1M		Amazon		TencentGR-1M		UserBehavior	
	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10
SASRec Kang & McAuley (2018)	0.0483	0.1010	0.0168	0.0414	0.4668	0.7153	0.2667	0.4188
BERT4Rec Sun et al. (2019)	0.0940	0.1912	0.0335	0.0743	0.2938	0.4711	0.4201	0.5455
RecBLR Liu et al. (2024a)	0.0793	0.1596	0.1918	0.2826	0.5992	0.8269	0.4914	0.6053
Mamba4Rec Liu et al. (2024b)	0.0588	0.1232	0.0418	0.0927	0.5956	0.8198	0.4087	0.5719
HSTU Zhai et al. (2024)	0.0906	0.1715	0.1201	0.2271	0.6911	0.8626	0.5155	0.6284
STAR-Rec Wang et al. (2025)	0.0903	0.1851	0.0879	0.1548	0.5099	0.7478	0.2880	0.4303
BlossomRec Ma et al. (2026)	0.1157	0.2137	0.1363	0.2234	0.6488	0.8276	0.4663	0.5886
TriScaleRec	**0.1234**	**0.2164**	**0.2096**	**0.2965**	**0.7327**	**0.8885**	**0.5468**	**0.6614**
Avg. Improv. (%)	+49.7%	+32.3%	+133.6%	+89.3%	+34.8%	+18.0%	+34.0%	+22.2%

† Avg. Improv. denotes the relative improvement of TriScaleRec over the average performance of all baseline methods.



(a)TencentGR-1M: NDCG@10 (b)TencentGR-1M: HR@10 (c)UserBehavior: NDCG@10 (d)UserBehavior: HR@10

Figure 5: Scaling performance of different sequential recommendation architectures across six model scales on TencentGR-1M and UserBehavior.

McAuley, 2016), Taobao UserBehavior (Alibaba Group, 2018), and TencentGR-1M (Pan et al., 2026). Detailed dataset descriptions and statistics are provided in Appendix A.

Baselines. We compare TriScaleRec with representative sequential recommendation methods, including SASRec (Kang & McAuley, 2018), BERT4Rec (Sun et al., 2019), HSTU (Zhai et al., 2024), BlossomRec (Ma et al., 2026), Mamba4Rec (Liu et al., 2024b), RecBLR (Liu et al., 2024a), and STAR-Rec (Wang et al., 2025). These baselines cover attention-based, state-space, recurrent, and hybrid sequence modeling paradigms. All methods use the same data preprocessing and evaluation protocol, with key hyperparameters selected on the validation set.

Evaluation Protocol. We formulate sequential recommendation as next-item prediction and evaluate all methods using HR@10 and NDCG@10 under the same sampled ranking protocol. Detailed evaluation settings are provided in Appendix B.

4.2 Overall Performance

Table 1 compares TriScaleRec with representative sequential recommendation methods on four benchmark datasets. TriScaleRec achieves the best performance across all eight evaluation metrics, demonstrating consistent effectiveness across datasets with substantially different scales and interaction densities. The improvements are particularly pronounced on Amazon, where TriScaleRec substantially outperforms existing methods on both NDCG@10 and HR@10.

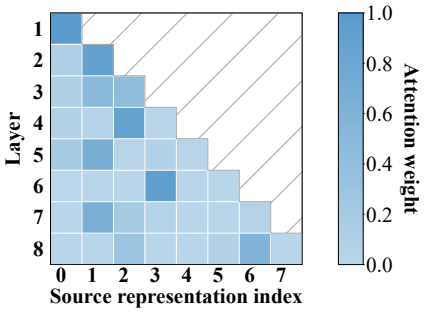
On the larger TencentGR-1M and UserBehavior datasets, TriScaleRec also consistently surpasses strong baselines such as HSTU and BlossomRec. These results suggest that the proposed scaling-oriented design remains effective as both the item space and interaction volume increase. We further investigate whether this advantage persists under increasing model scale in Section 4.3.

4.3 Scaling Behavior

We next investigate whether different architectures can consistently benefit from increasing model scale. For each scaling level, all models are configured with the same model width, number of layers, number of attention heads, and head dimension, while architecture-specific components are scaled according to their respective designs. We evaluate six progressively larger configurations to study how recommendation performance changes as model scale increases.

(a) Depth Scaling Performance

Layers	UserBehavior		TencentGR-1M	
	NDCG@10	HR@10	NDCG@10	HR@10
2	0.5388	0.6535	0.7147	0.8771
4	0.5427	0.6581	0.7281	0.8860
6	0.5449	0.6593	0.7299	0.8870
8	**0.5472**	**0.6612**	**0.7327**	**0.8885**



(b) Layer Representation Utilization

Figure 6: Analysis of Block Attention. (a) Performance scaling with increasing Transformer depth. (b) Layer utilization heatmap showing the contribution of different source depths.

As shown in Figure 5, the models exhibit markedly different scaling trajectories under the same structural scaling schedule. TriScaleRec maintains a smooth and monotonic improvement across all six scales on both datasets, indicating that additional depth and width continue to be effectively converted into recommendation quality even at larger model sizes. In contrast, the baselines reveal several characteristic scaling limitations: SASRec improves relatively steadily but with weaker marginal returns at larger scales, HSTU gradually approaches saturation, and BlossomRec exhibits more pronounced non-monotonic behavior, particularly on Taobao. These contrasting trajectories suggest that scaling in sequential recommendation is not an automatic consequence of adding parameters. Rather, the architecture determines how effectively newly introduced capacity can be utilized, and a scaling-oriented design should preserve positive marginal returns as the model grows.

4.4 Component Analysis

We conduct component-wise analyses to understand the contribution of each design in TriScaleRec. Specifically, we first perform factorized ablation experiments by individually removing Block Attention, LatentMoE, and Hybrid Muon/AdamW. We then further analyze each component to investigate how they improve depth utilization, capacity scaling, and optimization efficiency, respectively.

Factorized Ablation. As shown in Table 2, all three components contribute positively to the final performance. Removing any individual component leads to consistent performance degradation, demonstrating that Block Attention, LatentMoE, and Hybrid Muon/AdamW provide complementary improvements.

Table 2: Ablation study on TencentGR-1M.

Model	NDCG@10	HR@10
Dense Backbone	0.7079	0.8694
w/o Block Attention	0.7216	0.8821
w/o Muon	0.7176	0.8779
w/o LatentMoE	0.7273	0.8856
TriScaleRec	**0.7327**	**0.8885**

Block Attention Analysis. Block Attention is designed to improve the utilization of additional Transformer depth. We investigate whether newly added layers continue to provide useful representations as the model becomes deeper. Figure 6(a) reports the performance under different numbers of Transformer layers. As the depth increases from 2 to 8 layers, the model maintains consistent improvements, indicating that additional depth can be effectively utilized rather than becoming redundant. To further analyze the information flow across depths, Figure 6(b) visualizes the layer utilization patterns of Block Attention. The learned weights exhibit clear non-local and layer-specific dependencies, with later layers frequently retrieving representations from much earlier depths rather than relying on their immediate predecessors. This reveals that useful information is not confined to adjacent layers and supports viewing Block Attention as a learned depth-routing mechanism that dynamically selects where information should be retrieved from across the network hierarchy.

Table 3: Comparison of different feed-forward architectures on TencentGR-1M.

FFN Module	#Experts	Top-K	Total Params (M)	Active Params (M)	NDCG@10 / HR@10
Dense GatedMLP	1	1	12.583	12.583	0.7273 / 0.8856
Standard MoE	16	4	201.392	50.397	**0.7356 / 0.8885**
Standard MoE (Budget-Matched)	4	1	50.348	12.599	0.7270 / 0.8842
LatentMoE	16	4	51.446	13.697	0.7327 / 0.8885

LatentMoE Analysis. We further analyze whether LatentMoE (Elango et al., 2026) provides an effective capacity–computation trade-off. We first compare LatentMoE with dense and standard sparse MoE alternatives (Shazeer et al., 2017; Fedus et al., 2021) under different parameter budgets. We then study how its performance changes as the number of experts and the latent dimension increase.

As shown in Table 3, LatentMoE outperforms the budget-matched Standard MoE under a comparable total parameter budget, while approaching the performance of the substantially larger Standard MoE with far fewer active parameters. This comparison indicates that simply allocating more parameters to full-space experts is not necessarily the most efficient way to improve recommendation performance. By moving expert transformations into a compact latent space, LatentMoE enables a larger expert pool without proportionally increasing the computation activated for each input. In particular, compared with the full Standard MoE, LatentMoE preserves competitive recommendation quality while substantially reducing active parameter cost. These results demonstrate that LatentMoE provides a more favorable capacity–computation trade-off and supports the central design goal of expanding model capacity without proportionally increasing active computation.

Hybrid Muon/AdamW Analysis. We finally investigate how the assignment of Muon (Jordan et al., 2024) to different parameter groups affects optimization performance. Instead of uniformly applying Muon to all eligible parameters, we separately enable it for the attention modules, LatentMoE, and the feature tower, while keeping all remaining parameters optimized by AdamW (Loshchilov & Hutter, 2019).

Table 4: Muon assignment analysis on TencentGR-1M.

Configuration	NDCG@10	HR@10	Step Time (s) ↓
AdamW Only	0.7176	0.8779	0.776
Attention	0.7279	0.8859	0.780
LatentMoE	0.7233	0.8823	0.945
Feature Tower	0.7189	0.8795	0.775
Attention + LatentMoE	**0.7327**	0.8885	0.958
All Eligible Parameters	0.7317	**0.8890**	0.961

As shown in Table 4, the benefit of Muon strongly depends on the parameter group to which it is applied. Applying Muon to the attention parameters produces the largest gain among the individual groups, while applying it to the feature tower yields only a marginal improvement over AdamW. Combining Muon for Attention and LatentMoE further improves performance to a level comparable with applying Muon to all eligible parameters. Extending Muon more broadly provides no consistent additional benefit while slightly increasing per-step training cost. These results motivate our targeted Hybrid Muon/AdamW strategy.

5 Conclusion and Future Work

We present TriScaleRec, a scaling-oriented framework for sequential recommendation that improves how additional depth, capacity, and computation are translated into recommendation gains. Block Attention enhances cross-depth utilization, LatentMoE expands expert capacity with controlled active computation, and Hybrid Muon/AdamW provides targeted optimization for suitable parameter groups. Experiments on multiple benchmarks show that TriScaleRec achieves strong overall performance and more consistent scaling behavior than representative baselines. In particular, TriScaleRec maintains positive performance gains across the tested model scales, while several baselines exhibit saturation or fluctuations at larger configurations. These results suggest that effective scaling in sequential recommendation depends not only on increasing model size, but also on improving how additional model resources are utilized. Future work may further investigate these principles under larger datasets, longer sequences, and broader model scales.

USE OF AI ASSISTANTS

Generative AI tools were used solely for auxiliary purposes, including language polishing, grammar correction, improving presentation clarity, and assisting with LaTeX formatting. The core research ideas, technical claims, methodological design, theoretical arguments, experimental conclusions, and overall scientific viewpoints presented in this work were independently developed by the authors. All AI-assisted edits were carefully reviewed and verified by the authors, who take full responsibility for the final content of the paper.

REFERENCES

Alibaba Group. User behavior data from Taobao for recommendation. Tianchi Dataset, 2018. URL https://tianchi.aliyun.com/dataset/649.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Gary Cottrell, and Julian J. McAuley. ReZero is all you need: Fast convergence at large depth. In *UAI*, pp. 1352–1361, 2021.

Zhimin Chen, Chenyu Zhao, Ka Mo, Yunjiang Jiang, Jane Lee, Khushhall Chandra Mahajan, Ning Jiang, Kai Ren, Charlie Li, and Wen-Yun Yang. Massive memorization with hundreds of trillions of parameters for sequential transducer generative recommenders. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/41bba7b0f5c81e789a20bb16a370aeeb-Abstract-Conference.html.

Wenzhuo Cheng, Shipeng Nie, Qixin Guo, Xuefeng Sun, Jianguo Lou, and Zhengwei Zheng. DeRes: Decoupling residual stability and adaptivity for scalable CTR prediction. *CoRR*, abs/2606.07980, 2026.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. *CoRR*, abs/2401.06066, 2024.

Venmugil Elango, Nidhi Bhatia, Roger Waleffe, Rasoul Shafipour, Tomer Asida, Abhinav Khattar, Nave Assaf, Maximilian Golub, Joey Guman, Tiyasa Mitra, Ritchie Zhao, Ritika Borkar, Ran Zilberstein, Mostofa Patwary, Mohammad Shoeybi, and Bita Rouhani. LatentMoE: Toward optimal accuracy per FLOP and parameter in mixture of experts. *CoRR*, abs/2601.18089, 2026.

Xinyan Fan, Zheng Liu, Jianxun Lian, Wayne Xin Zhao, Xing Xie, and Ji-Rong Wen. Lighter and better: Low-rank decomposed self-attention networks for next-item recommendation. In *SIGIR*, pp. 1733–1737, 2021.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021.

F. Maxwell Harper and Joseph A. Konstan. The MovieLens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pp. 507–517, 2016.

Bal'azs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pp. 2261–2269, 2017.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206, 2018.

Kimi Team. Attention residuals. *CoRR*, abs/2603.15031, 2026.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. GShard: Scaling giant models with conditional computation and automatic sharding. *CoRR*, abs/2006.16668, 2020.

Jiacheng Li, Yujie Wang, and Julian J. McAuley. Time interval aware self-attention for sequential recommendation. In *WSDM*, pp. 322–330, 2020.

Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *CIKM*, pp. 1419–1428, 2017.

Jingyu Li, Zhaocheng Du, Qianhui Zhu, Kaiyuan Li, Zhicheng Zhang, Song-Li Wu, Chaolang Li, and Pengwen Dai. Collectivekv: Decoupling and sharing collaborative information in sequential recommendation. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4078c8b648dc107aedbdf561dd4edc2a-Abstract-Conference.html.

Chengkai Liu, Jianghao Lin, Hanzhou Liu, Jianling Wang, and James Caverlee. Behavior-dependent linear recurrent units for efficient sequential recommendation. In *CIKM*, pp. 1430–1440, 2024a.

Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. Mamba4Rec: Towards efficient sequential recommendation with selective state space models. *CoRR*, abs/2403.03900, 2024b.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *ICLR*, 2024c. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/06960915ba8674c7a898ec0b472b80ff-Abstract-Conference.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Mengyang Ma, Xiaopeng Li, Wanyu Wang, Zhaocheng Du, Jingtong Gao, Pengyue Jia, Yuyang Ye, Yiqi Wang, Yunpeng Weng, Weihong Luo, Xiao Han, and Xiangyu Zhao. BlossomRec: Block-level fused sparse attention mechanism for sequential recommendations. In *WWW*, pp. 6389–6399, 2026.

Junwei Pan et al. Tencent advertising algorithm challenge 2025: All-modality generative recommendation. *CoRR*, abs/2604.04976, 2026.

Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. Contrastive learning for representation degeneration problem in sequential recommendation. In *WSDM*, pp. 813–823, 2022.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Comput. Surv.*, 51(4):66:1–66:36, 2018.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pp. 811–820, 2010.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pp. 1441–1450, 2019.

Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pp. 565–573, 2018.

A"aron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *TPAMI*, 46(10):6761–6774, 2024.

Maolin Wang, Sheng Zhang, Ruocheng Guo, Wanyu Wang, Xuetao Wei, Zitao Liu, Hongzhi Yin, Yi Chang, and Xiangyu Zhao. STAR-Rec: Making peace with length variance and pattern diversity in sequential recommendation. In *SIGIR*, pp. 1530–1540, 2025.

Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. Sequential recommender systems: Challenges, progress and prospects. In *IJCAI*, pp. 6332–6338, 2019.

Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. Contrastive learning for sequential recommendation. In *ICDE*, pp. 1259–1273, 2022.

Wujiang Xu, Qitian Wu, Zujie Liang, Jiaojiao Han, Xuying Ning, Yunxiao Shi, Wenfang Lin, and Yongfeng Zhang. Slmrec: Distilling large language models into small for sequential recommendation. In *ICLR*, 2025. URL https://openreview.net/forum?id=G4wARwjF8M.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*, pp. 58484–58509, 2024.

Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. Scaling law of large sequential recommendation models. In *RecSys*, pp. 444–453, 2024.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, pp. 1893–1902, 2020.

Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced MLP is all you need for sequential recommendation. In *WWW*, pp. 2388–2399, 2022.

Pablo Zivic, Hernan Vazquez, and Jorge Sánchez. Scaling sequential recommendation models with transformers. In *SIGIR*, pp. 1567–1577, 2024.

A Dataset Statistics

We provide detailed descriptions of the four evaluation datasets below. Their statistics after preprocessing are summarized in Table 5.

- **ML-1M** (Harper & Konstan, 2015) is a widely used benchmark for sequential recommendation based on user ratings of movies. After preprocessing, it contains 6,040 users, 3,706 items, and approximately 1.0 million interactions. With an average sequence length of 165.00, ML-1M represents a relatively small but dense recommendation setting, providing a useful benchmark for evaluating sequential modeling under long and information-rich user histories.
- **Amazon** (He & McAuley, 2016) contains user–product interactions collected from the Amazon platform. The processed dataset contains approximately 3.04 million users, 1.57 million items, and 41.1 million interactions. Its average sequence length is only 13.55, which is substantially shorter than those of the other datasets. Amazon therefore provides a large but relatively sparse setting for evaluating recommendation models when limited historical interactions are available for each user.

Table 5: Statistics of the evaluation datasets.

Dataset	#Users	#Items	#Interactions	Avg. Length
ML-1M	6,040	3,706	1.0M	165.00
Amazon	3.04M	1.57M	41.1M	13.55
Taobao UserBehavior	0.99M	4.16M	100.1M	101.37
TencentGR-1M	1.00M	4.78M	90.2M	90.06

- **Taobao UserBehavior** (Alibaba Group, 2018) is a large-scale e-commerce dataset containing sequential user behaviors on the Taobao platform. After preprocessing, it includes approximately 0.99 million users, 4.16 million items, and 100.1 million interactions, with an average sequence length of 101.37. Its large item space and dense behavioral histories make it suitable for evaluating sequential models under large-scale and interaction-rich recommendation settings.
- **TencentGR-1M** (Pan et al., 2026) is a large-scale recommendation dataset consisting of user–item interaction sequences. The processed dataset contains approximately 1.00 million users, 4.78 million items, and 90.2 million interactions, with an average sequence length of 90.06. Among the four benchmarks, TencentGR-1M has the largest item space while maintaining relatively long user histories, providing a challenging setting for evaluating large-scale sequential recommendation.

Overall, the four datasets exhibit substantial differences in user scale, item-space size, interaction density, and sequence length. They therefore provide complementary evaluation settings ranging from compact and dense recommendation to sparse and large-scale sequential modeling.

B Implementation Details

B.1 Baseline Details

We provide additional details of the baseline methods used in the experiments. The selected methods cover representative sequence modeling paradigms and recent architectures for sequential recommendation.

- **SASRec** (Kang & McAuley, 2018) is a self-attentive sequential recommendation model that uses causal self-attention to capture dependencies among historical interactions and predict the next item.
- **BERT4Rec** (Sun et al., 2019) adopts a bidirectional Transformer encoder and masked item prediction objective to learn contextualized representations of user behavior sequences.
- **HSTU** (Zhai et al., 2024) is a large-scale sequential transduction architecture designed for recommendation, with particular emphasis on scalable modeling of long user behavior sequences.
- **BlossomRec** (Ma et al., 2026) introduces block-level fused sparse attention to improve the efficiency and expressiveness of sequential recommendation.
- **Mamba4Rec** (Liu et al., 2024b) adapts selective state-space modeling to sequential recommendation, providing an alternative to Transformer-based sequence encoders.
- **RecBLR** (Liu et al., 2024a) models sequential dynamics using behavior-dependent linear recurrent units, aiming to improve both efficiency and sequence representation quality.
- **STAR-Rec** (Wang et al., 2025) combines complementary sequence modeling components to handle sequence-length variation and diverse behavioral patterns in sequential recommendation.

Together, these baselines provide coverage of attention-based, state-space, recurrent, and hybrid architectures, enabling a broad comparison with TriScaleRec under the same preprocessing and evaluation settings.

Table 6: Common training and evaluation settings.

Setting	Value
Data split	Train/Validation/Test = 8:1:1
Maximum sequence length	100
Training random seed	42
Validation/test candidate seed	5
Validation negatives	1,000
Test negatives	10,000
Evaluation candidate pool	Fixed global candidate pool
Model selection	Best validation NDCG@10
Mixed precision	BF16 AMP
Gradient clipping	Max norm = 1.0
Distributed training	4-GPU DDP
Evaluation frequency	Every epoch
Early-stop minimum delta	0
Evaluation metrics	NDCG@10, HR@10

B.2 Training Details

All models are implemented in PyTorch and trained using 4-GPU distributed data parallelism (DDP) on NVIDIA A800 80GB GPUs. We adopt a chronological 8:1:1 split for training, validation, and testing. Model selection is based on validation NDCG@10, and the checkpoint with the best validation performance is used for final evaluation. The common training and evaluation settings are summarized in Table 6.

We formulate sequential recommendation as next-item prediction and report NDCG@10 and HR@10. During validation, each ground-truth item is ranked against 1,000 sampled negatives, while testing uses 10,000 sampled negatives. For ML-1M, whose item vocabulary contains only 3,706 items, we instead perform full-ranking evaluation over the complete item set. The candidate sets are generated using a fixed random seed of 5, and a fixed global candidate pool is shared across evaluation batches so that candidate construction is independent of batch size. To maintain a consistent configuration across the six scaling levels and reserve GPU memory for larger models, we use 102,400 training negatives and an embedding dimension of 64 in the main experiments. Larger configurations are examined separately in E.

For TriScaleRec, unless otherwise specified, we use an 8-layer Transformer with 8 attention heads and a learning rate of 6.5×10^{-4}. LatentMoE contains 16 experts with a latent dimension of 128 and Top-4 routing. The model is trained using the Hybrid Muon/AdamW optimizer described in Section 3.4 and the InfoNCE objective described in Section 3.5.

For the larger SASRec configurations used in the scaling experiments, directly extending the original setup to greater width and depth leads to unstable convergence rather than primarily causing memory limitations. In particular, Post-Norm, unscaled residual connections, and the original learning rate amplify activation and gradient fluctuations as model scale increases. We therefore adopt a stabilized configuration at the largest scales by replacing Post-Norm with Pre-Norm, applying a residual scaling factor of 0.5, reducing the learning rate to 1×10^{-4}, extending training to 20 epochs, and increasing the early-stopping patience. These modifications are introduced solely to ensure stable convergence at larger model scales while preserving the underlying SASRec architecture.

B.3 Scaling Configurations

To ensure a fair scaling comparison, we construct six model scales for SASRec, HSTU, BlossomRec, and TriScaleRec. We conduct the scaling analysis on TencentGR-1M and Taobao UserBehavior, as these datasets provide substantially larger user populations, item spaces, and interaction volumes than ML-1M, while exhibiting different levels of sequence sparsity and interaction density. Such diverse large-scale settings provide a more appropriate testbed for examining whether perfor-

Table 7: Detailed model configurations used for the six scaling levels.

Level	Model	Width	Layers	Heads	Head Dim	FFN / Expert Dim	Other Configuration
L1	SASRec	64	2	2	32	64	Post-LN
	HSTU	64	2	2	32	linear_dim=32	attention_dim=32
	BlossomRec	64	2	2	32	inner=128	KV Heads=2
	TriScaleRec	64	2	2	32	expert_dim=16	LatentMoE
L2	SASRec	128	4	4	32	128	Post-LN
	HSTU	128	4	4	32	linear_dim=32	attention_dim=32
	BlossomRec	128	4	4	32	inner=256	KV Heads=2
	TriScaleRec	128	4	4	32	expert_dim=32	LatentMoE
L3	SASRec	256	8	8	32	256	Post-LN
	HSTU	256	8	8	32	linear_dim=32	attention_dim=32
	BlossomRec	256	8	8	32	inner=512	KV Heads=2
	TriScaleRec	256	8	8	32	expert_dim=64	LatentMoE
L4	SASRec	512	16	16	32	512	Post-LN
	HSTU	512	16	16	32	linear_dim=32	attention_dim=32
	BlossomRec	512	16	16	32	inner=1024	KV Heads=2
	TriScaleRec	512	16	16	32	expert_dim=128	LatentMoE
L5	SASRec	768	24	24	32	768	Pre-LN, Residual Scale=0.5
	HSTU	768	24	24	32	linear_dim=32	attention_dim=32
	BlossomRec	768	24	24	32	inner=1536	KV Heads=2
	TriScaleRec	768	24	24	32	expert_dim=192	LatentMoE
L6	SASRec	1024	32	32	32	1024	Pre-LN, Residual Scale=0.5
	HSTU	1024	32	32	32	linear_dim=32	attention_dim=32
	BlossomRec	1024	32	32	32	inner=2048	KV Heads=2
	TriScaleRec	1024	32	32	32	expert_dim=256	LatentMoE

mance gains can be sustained as model capacity increases. The model width, number of layers, and number of attention heads are progressively increased with the scaling level, while architecture-specific dimensions are adjusted accordingly. Detailed configurations are summarized in Table 7.

C Algorithmic Details

We summarize the training procedure of TriScaleRec in Algorithm 1, highlighting the interaction among Block Attention, LatentMoE, and Hybrid Muon/AdamW optimization.

D Active-Parameter-Matched Scaling

As a complementary scaling analysis, we further evaluate the compared models under strictly matched active-parameter budgets. Unlike the main scaling experiment, where models at the same level share the same width, number of layers, and attention heads, this setting adjusts architecture-specific configurations so that the number of active encoder parameters per token is approximately aligned across models. We construct six levels ranging from about 2.5M to 80M active parameters and exclude dataset-dependent embedding tables from the parameter count. This experiment provides an additional view of scaling behavior under a more tightly controlled computational-capacity budget and helps assess whether the observed scaling trends remain consistent when active model size is explicitly matched.

Across both datasets and evaluation metrics, TriScaleRec consistently delivers the strongest performance throughout the scaling process. Its performance improves steadily as model capacity increases, demonstrating stable and effective scalability. In contrast, the competing methods exhibit noticeable fluctuations, early saturation, or even performance degradation at certain scales, suggesting that simply enlarging conventional architectures does not always translate into better recommendation quality. The consistent upward trend of TriScaleRec indicates that its architectural design can

Algorithm 1 Training Procedure of TriScaleRec

Input: Training sequences $\mathcal{D}$, number of layers L, block size S, number of experts E, Top-K value K
Output: Trained model parameters Θ

Partition Θ into Θ_{Muon} and Θ_{AdamW}
while *not converged* **do**
  Sample a mini-batch $(\mathcal{S}, \mathcal{Y})$ from $\mathcal{D}$
  $H^{(0)} \leftarrow \text{Embed}(\mathcal{S})$
  $\mathcal{M} \leftarrow \{H^{(0)}\}$
  for $l = 1$ to L **do**
    Determine the current Fusion Block b
    Construct $\mathcal{D}_{b,l}$ by equation 2
    Compute $\widetilde{F}_{b,l+1}$ by equation 3–equation 4
    Apply causal self-attention to obtain x_t
    Compute routing probabilities p_t by equation 6
    Project x_t into the latent space by equation 5
    Select Top-K experts and compute $\hat{z}_t$ by equation 7
    Recover y_t by equation 8
    if $l \bmod S = 0$ **then**
      Store the completed block representation in $\mathcal{M}$
    end if
  end for
  Compute the next-item InfoNCE loss $\mathcal{L}_{\text{InfoNCE}}$
  Backpropagate $\nabla_{\Theta}\mathcal{L}_{\text{InfoNCE}}$
  Update Θ_{Muon} with Muon and Θ_{AdamW} with AdamW
end while
return Θ

Table 8: Active-parameter budgets used for the six scaling levels.

Level	Target	SASRec	HSTU	BlossomRec	TriScaleRec
L1	∼2.5M	2.418	2.462	2.437	2.516
L2	∼5M	5.035	4.980	4.862	5.045
L3	∼10M	9.857	9.917	10.184	10.006
L4	∼20M	20.405	19.982	20.320	20.412
L5	∼40M	39.998	39.372	40.424	38.543
L6	∼80M	80.419	79.642	79.127	77.754

more effectively exploit additional model capacity, while maintaining robust performance across datasets with different characteristics. These results highlight the superior scalability, stability, and generalization ability of TriScaleRec.

E Additional Ablation Experiments

E.1 Effect of Feature Embedding Dimension

We further examine the sensitivity of TriScaleRec to the feature embedding dimension on both TencentGR-1M and UserBehavior. We vary the embedding dimension from 64 to 512 while keeping the remaining model configuration unchanged. The results are reported in Table 9.

Across both datasets, increasing the embedding dimension initially improves recommendation performance, suggesting that a sufficiently expressive embedding space is important for representing user–item interactions. However, the gains gradually saturate as the embedding dimension increases, and the optimal configuration differs across datasets: TencentGR-1M achieves its best performance with a larger embedding dimension, whereas UserBehavior reaches its optimum at a more moderate dimension. This difference suggests that the preferred embedding capacity may depend on dataset

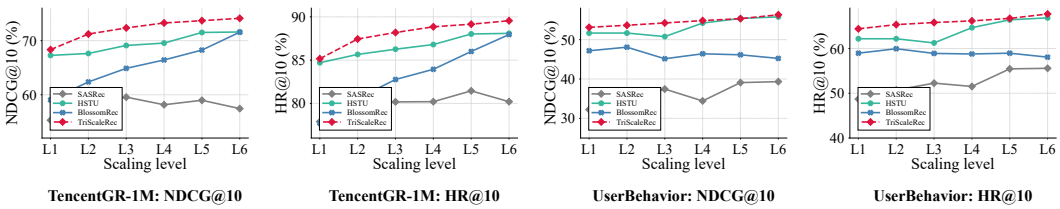


Figure 7: Scaling performance of different sequential recommendation architectures under matched active-parameter budgets on TencentGR-1M and UserBehavior.

Table 9: Effect of the feature embedding dimension on TencentGR-1M and UserBehavior.

Embed. Dim.	TencentGR-1M		UserBehavior	
	NDCG@10 ↑	HR@10 ↑	NDCG@10 ↑	HR@10 ↑
64(Default)	73.27	88.85	54.72	66.12
128	75.24	89.92	57.06	68.10
256	75.36	90.00	**57.09**	**68.38**
512	**75.41**	**90.12**	56.93	68.32

characteristics such as the scale of the item space, interaction volume, and representation complexity. In particular, larger or more diverse item spaces may benefit from higher-dimensional embeddings, while further increasing the embedding dimension can become redundant once the required representation capacity is reached. Overall, these results indicate that the embedding dimension should be adapted to the characteristics and scale of the dataset rather than uniformly increased with model size.

E.2 Effect of Block Size

To isolate the effect of Block Attention and avoid potential interference from other components, we conduct this experiment using a simplified variant of TriScaleRec that retains only Block Attention, while disabling LatentMoE and the Hybrid Muon/AdamW optimization strategy. We then vary the block size, i.e., the number of consecutive Transformer blocks grouped into each fusion unit, over $\{1, 2, 3, 6\}$. The results are reported in Table 10.

Across both datasets, a block size of 1 achieves the best recommendation performance, reaching NDCG@10/HR@10 of 0.1204/0.2147 on ML-1M and 0.7216/0.8908 on TencentGR-1M. Larger block sizes generally yield lower scores, although performance among sizes 2, 3, and 6 is not strictly monotonic. For example, increasing the block size from 1 to 6 reduces NDCG@10 from 0.7216 to 0.6645 on TencentGR-1M. These results are consistent with finer-grained fusion allowing Block Attention to distinguish and reuse representations from different depths more effectively. We therefore use a block size of 1 as the default configuration in TriScaleRec.

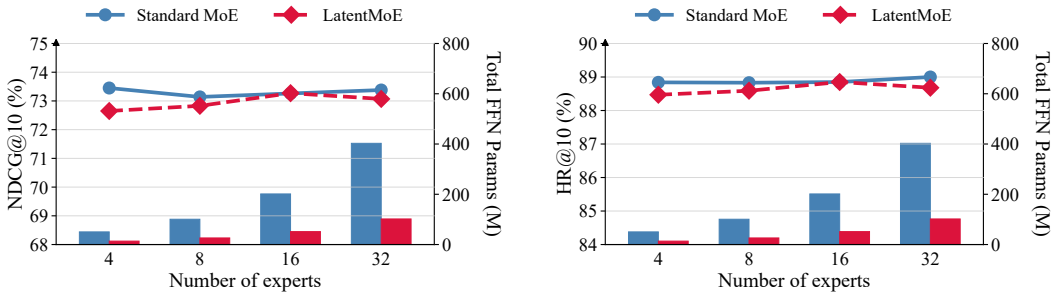
E.3 Additional Analysis of LatentMoE

Effect of the Number of Experts. We vary the number of experts while keeping the number of activated experts fixed, and compare LatentMoE with Standard MoE under the same routing setting. This experiment examines whether LatentMoE can increase total expert capacity without proportionally increasing active computation.

As shown in Figure 8, LatentMoE maintains competitive recommendation performance as the number of experts increases, while its active parameter cost remains nearly unchanged because only a fixed number of experts are activated for each input. In contrast, Standard MoE requires substantially more active FFN parameters because expert computation is performed directly in the full hidden space. In particular, with 16 experts, LatentMoE achieves performance close to Standard MoE while using only about one-quarter of its active FFN parameters. These results indicate that LatentMoE can expand total expert capacity more efficiently by decoupling the number of available experts from the amount of active computation.

Table 10: Effect of different Block Attention block sizes on recommendation performance.

Block size	ML-1M		TencentGR-1M	
	NDCG@10	HR@10	NDCG@10	HR@10
1	**0.1204**	**0.2147**	**0.7216**	**0.8908**
2	0.1182	0.2123	0.6665	0.8653
3	0.1128	0.2134	0.6673	0.8664
6	0.1172	0.2129	0.6645	0.8660



(a) NDCG@10 (b) HR@10

Figure 8: Effect of the number of experts in LatentMoE. Performance is compared with Standard MoE as the total expert capacity increases.

Effect of the Latent Dimension. We next investigate the effect of the latent dimension, which controls the compression ratio and computational cost of expert computation in LatentMoE. We vary the latent dimension from 64 to 512 while keeping the number of experts and Top-K routing fixed at 16 and 4, respectively. Table 11 reports the resulting parameter cost and recommendation performance.

As shown in Table 11, recommendation performance does not improve monotonically as the latent dimension increases. This suggests that the effectiveness of LatentMoE is not determined solely by the representational capacity of the latent space. Since only a subset of experts is activated for each input through sparse routing, enlarging the latent dimension also changes the representations on which expert selection and specialization are performed, which may lead to different routing and optimization behaviors across configurations. Consequently, a larger latent space does not necessarily translate into better recommendation performance. These results indicate that the latent dimension should be viewed as part of a joint capacity–routing trade-off rather than as an isolated scaling factor. In practice, a moderate latent dimension provides sufficient representational capacity while preserving the computational advantages of latent-space expert computation, motivating the use of 128 as the default setting in TriScaleRec.

E.4 Effect of Training Negative Samples

We further investigate how the number of training negatives affects the performance of TriScaleRec. Specifically, we vary the size of the sampled negative pool during training while keeping the remaining configurations unchanged. The results on TencentGR-1M and Taobao are reported in Table 12.

As shown in Table 12, the effect of increasing the number of training negatives differs across datasets. On TencentGR-1M, performance improves substantially when the negative pool is increased beyond 102,400, with the best NDCG@10 and HR@10 obtained using 409,600 negatives. This suggests that a larger negative pool can provide richer contrastive signals and improve item discrimination on this dataset.

The trend on Taobao is less monotonic. Increasing the number of negatives from the default 102,400 to 204,800 substantially improves NDCG@10, whereas further increasing it to 409,600 provides no additional benefit. HR@10 shows a similar saturation pattern, with the highest value achieved using

Table 11: Effect of the latent dimension in LatentMoE on TencentGR-1M.

Latent Dim.	Total FFN Params	Active FFN Params	NDCG@10 ↑	HR@10 ↑
64	25.756M	6.881M	72.66	88.45
128(Default)	51.446M	13.697M	73.27	88.85
256	102.826M	27.329M	72.93	88.73
512	205.586M	54.591M	**73.35**	**88.89**

Table 12: Effect of the number of training negatives on TencentGR-1M and Taobao.

Train Negatives	TencentGR-1M		Taobao UserBehavior	
	NDCG@10	HR@10	NDCG@10	HR@10
51,200	0.7329	0.8910	0.5670	**0.6780**
102,400	0.7327	0.8885	0.5472	0.6612
204,800	0.7565	0.9005	**0.5720**	0.6778
409,600	**0.7587**	**0.9028**	0.5694	0.6745

† The 102,400-negative setting is used as the default configuration for Taobao in the main experiments.

51,200 negatives and only marginal differences among the larger settings. These results indicate that simply increasing the negative pool does not guarantee consistent performance gains.

Overall, the experiment suggests that the appropriate number of training negatives is dataset dependent. Increasing the negative pool can strengthen the contrastive learning signal, but the resulting gains eventually saturate and may become non-monotonic. This further highlights the importance of considering training configuration, in addition to model size, when scaling sequential recommendation models.

E.5 Evaluation under Different Negative Sampling Sizes

To investigate whether the performance gains of TriScaleRec are sensitive to the negative sampling strategy, we further conduct an additional evaluation using 1,000 sampled negatives during testing. The training procedure and all other evaluation settings remain identical to the main experiments, except for the number of candidate negatives.

As shown in Table 13, TriScaleRec consistently maintains the strongest performance across different datasets under the reduced candidate setting. The relative ranking among representative sequential recommendation models remains stable, indicating that the improvements of TriScaleRec are not attributed to a specific negative sampling configuration. Instead, the results further demonstrate that the proposed scaling-oriented designs provide consistent benefits under different evaluation difficulties.

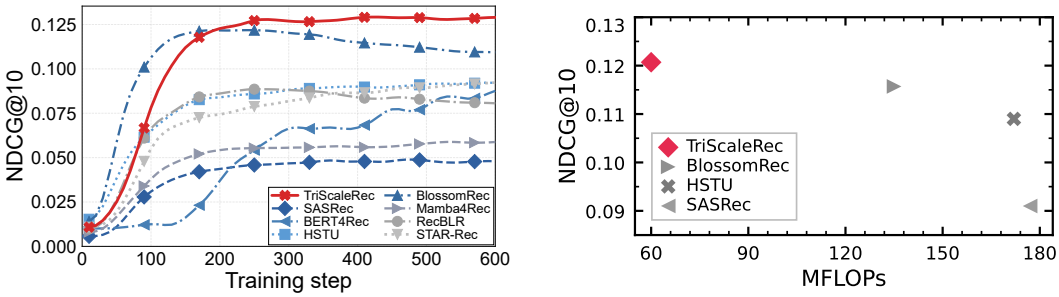
F Additional Efficiency Analysis

We provide additional analysis of the convergence behavior and performance–efficiency trade-off of TriScaleRec. For convergence, we compare validation NDCG@10 against SASRec (Kang & McAuley, 2018), HSTU (Zhai et al., 2024), and BlossomRec (Ma et al., 2026) over training steps. For computational efficiency, we compare recommendation performance with inference cost under the best-performing configuration of each model. All methods follow the same data preprocessing and evaluation protocol.

As shown in Figure 9(a), TriScaleRec improves rapidly during early training and remains stable after convergence, while the compared baselines exhibit slower progress or lower converged performance. This provides additional evidence that the proposed design can be effectively optimized as model capacity increases.

Table 13: Performance comparison under evaluation with 1,000 sampled negatives. Best results are highlighted in bold with gray backgrounds, and second-best results are underlined with dashed lines.

Model	ML-1M		Amazon		TencentGR-1M		Taobao UserBehavior	
	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10
SASRec (Kang & McAuley, 2018)	0.1424	0.2821	0.1237	0.2216	0.7965	0.9478	0.5372	0.7298
BERT4Rec (Sun et al., 2019)	0.2175	0.3812	0.1582	0.2381	0.5986	0.7909	0.6224	0.7480
RecBLR (Liu et al., 2024a)	0.1897	0.3538	0.3674	0.4970	0.8682	0.9539	0.6750	0.7825
Mamba4Rec (Liu et al., 2024b)	0.1556	0.3091	0.1888	0.2803	0.8629	0.9631	0.6572	0.7960
HSTU (Zhai et al., 2024)	0.2018	0.3639	0.3298	0.4481	0.8923	0.9676	0.6811	0.7580
TriScaleRec	**0.2316**	**0.3978**	**0.3995**	**0.5273**	**0.9253**	**0.9769**	**0.7246**	**0.8224**



(a) Convergence

(b) Performance–compute trade-off

Figure 9: Additional efficiency analysis on ML-1M. (a) Validation NDCG@10 over training steps. (b) NDCG@10 versus inference MFLOPs per sequence.

Figure 9(b) further shows that TriScaleRec achieves a favorable recommendation–computation trade-off, attaining higher NDCG@10 at a competitive inference cost. Together, these results suggest that the performance gains of TriScaleRec are associated with more effective utilization of model capacity rather than simply increased computation.

G Theoretical Analysis and Proofs

We provide formal results for the two architectural components of TriScaleRec. We first show that Block Attention can recover conventional adjacent-layer propagation with a controllable approximation error. We then characterize its direct cross-depth information path through the Jacobian of the fusion operator. Finally, we analyze the approximation and active-computation properties of LatentMoE.

G.1 Block Attention as a Generalization of Adjacent Propagation

Let $X \in \mathbb{R}^{n\times d}$ denote an input sequence and consider a Transformer consisting of B blocks. Conventional adjacent-layer propagation is written as

$$H^{(0)} = X, \qquad H^{(b+1)} = \Phi_b\left(H^{(b)}\right), \quad b = 0, \dots, B-1, \tag{14}$$

where Φ_b denotes the transformation implemented by the b-th Transformer block.

For Block Attention, the representation at position t of target block b is obtained from all accessible depth representations:

$$\begin{aligned} e_{b,t,j} &= q_b^\top R\left(\widehat{h}_t^{(j)}\right), \\ \alpha_{b,t,j} &= \frac{\exp(e_{b,t,j})}{\sum_{k=0}^{b}\exp(e_{b,t,k})}, \\ z_{b,t} &= \sum_{j=0}^{b} \alpha_{b,t,j}\widehat{h}_t^{(j)}, \qquad \widehat{H}^{(b+1)} = \Phi_b(Z_b), \end{aligned} \tag{15}$$

where $R(\cdot)$ denotes RMSNorm and $Z_b = [z_{b,1}, \ldots, z_{b,n}]^\top$.

Our analysis assumes bounded intermediate representations, Lipschitz Transformer blocks, and query separability of the immediately preceding representation:

$$\|\widehat{h}_t^{(j)}\|_2 \le M, \qquad \|\Phi_b(A) - \Phi_b(B)\|_F \le L_b \|A - B\|_F, \qquad e_{b,t,b} - \max_{j<b} e_{b,t,j} \ge \gamma_b > 0. \tag{16}$$

Lemma 1 (Softmax leakage)**.** *Under equation 16, the total attention mass assigned to non-adjacent representations satisfies*

$$1 - \alpha_{b,t,b} \le \frac{be^{-\gamma_b}}{1 + be^{-\gamma_b}} \le be^{-\gamma_b}. \tag{17}$$

Proof. From the margin condition in equation 16, for every $j < b$,

$$e_{b,t,j} - e_{b,t,b} \le -\gamma_b,$$

and therefore $\exp(e_{b,t,j} - e_{b,t,b}) \le e^{-\gamma_b}$.

Dividing the numerator and denominator of $\alpha_{b,t,b}$ by $\exp(e_{b,t,b})$ gives

$$\begin{aligned} \alpha_{b,t,b} &= \frac{1}{1 + \sum_{j=0}^{b-1} \exp(e_{b,t,j} - e_{b,t,b})} \\ &\ge \frac{1}{1 + be^{-\gamma_b}}. \end{aligned} \tag{18}$$

Hence,

$$\begin{aligned} 1 - \alpha_{b,t,b} &\le 1 - \frac{1}{1 + be^{-\gamma_b}} \\ &= \frac{be^{-\gamma_b}}{1 + be^{-\gamma_b}} \le be^{-\gamma_b}, \end{aligned} \tag{19}$$

which proves equation 17. □

The leakage bound further controls the discrepancy between Block Attention fusion and conventional adjacent propagation. For each position t,

$$\begin{aligned} \left\| z_{b,t} - \widehat{h}_t^{(b)} \right\|_2 &= \left\| \sum_{j=0}^{b-1} \alpha_{b,t,j} \left(\widehat{h}_t^{(j)} - \widehat{h}_t^{(b)} \right) \right\|_2 \\ &\le \sum_{j=0}^{b-1} \alpha_{b,t,j} \left\| \widehat{h}_t^{(j)} - \widehat{h}_t^{(b)} \right\|_2 \\ &\le 2M \left(1 - \alpha_{b,t,b}\right) \le 2Mbe^{-\gamma_b}. \end{aligned} \tag{20}$$

Summing over all sequence positions yields

$$\left\| Z_b - \widehat{H}^{(b)} \right\|_F \le 2M\sqrt{n}\, be^{-\gamma_b}. \tag{21}$$

Theorem 3 (Approximation of adjacent-layer propagation)**.** *Let F_{Adj} denote a B-block Transformer using equation 14, and let F_{BA} denote the corresponding Block-Attention model using the same Transformer operators. Under equation 16,*

$$\boxed{\|F_{\mathrm{BA}}(X) - F_{\mathrm{Adj}}(X)\|_F \le 2M\sqrt{n} \sum_{b=1}^{B-1} be^{-\gamma_b} \prod_{r=b}^{B-1} L_r.} \tag{22}$$

Proof. Define the approximation error at depth b as

$$E_b = \left\| \widehat{H}^{(b)} - H^{(b)} \right\|_F.$$

Since the two models receive the same input representation, $E_0 = 0$.

Using equation 14 and equation 15, together with the Lipschitz condition in equation 16, we obtain

$$\begin{aligned} E_{b+1} &= \left\| \Phi_b(Z_b) - \Phi_b(H^{(b)}) \right\|_F \\ &\le L_b \left\| Z_b - H^{(b)} \right\|_F \\ &\le L_b \left(\left\| Z_b - \widehat{H}^{(b)} \right\|_F + \left\| \widehat{H}^{(b)} - H^{(b)} \right\|_F \right) \\ &\le L_b \left(E_b + 2M\sqrt{n}\, b e^{-\gamma_b} \right). \end{aligned} \tag{23}$$

Recursively expanding equation 23 from $b = 0$ to $B - 1$, and noting that the $b = 0$ fusion term vanishes, gives

$$E_B \le 2M\sqrt{n} \sum_{b=1}^{B-1} b e^{-\gamma_b} \prod_{r=b}^{B-1} L_r. \tag{24}$$

Since $E_B = \|F_{\mathrm{BA}}(X) - F_{\mathrm{Adj}}(X)\|_F$, equation 22 follows. □

If $L_b \le \Lambda$ and $\gamma_b \ge \gamma$ for every block, then

$$\|F_{\mathrm{BA}}(X) - F_{\mathrm{Adj}}(X)\|_F \le 2M\sqrt{n}\, e^{-\gamma} \sum_{b=1}^{B-1} b\Lambda^{B-b} \xrightarrow[\gamma\to\infty]{} 0. \tag{25}$$

Consequently, conventional adjacent propagation is contained in the closure of the Block-Attention function family:

$$\boxed{\mathcal{F}_{\mathrm{Adj}} \subseteq \overline{\mathcal{F}_{\mathrm{BA}}}.} \tag{26}$$

G.2 Cross-Depth Information Preservation

The previous result establishes recoverability of the original backbone. We next analyze the additional direct information paths introduced by Block Attention. For clarity, we suppress block and position indices and write

$$z = \sum_{k=0}^{b} \alpha_k s_k, \qquad \alpha_k = \frac{\exp(q^\top R(s_k))}{\sum_m \exp(q^\top R(s_m))}. \tag{27}$$

Lemma 2 (Exact Block-Attention Jacobian)**.** *Let $g_j = \nabla_{s_j}(q^\top R(s_j))$. The Jacobian of the fused representation with respect to an accessible depth representation s_j is*

$$\boxed{\frac{\partial z}{\partial s_j} = \alpha_j \left[I + (s_j - z) g_j^\top \right].} \tag{28}$$

Proof. Let $e_k = q^\top R(s_k)$ and $g_j = \nabla_{s_j} e_j$. Since e_k depends only on s_k, the softmax derivative gives

$$\nabla_{s_j} \alpha_k = \alpha_k \left(\mathbb{I}[k = j] - \alpha_j \right) g_j.$$

Differentiating $z = \sum_k \alpha_k s_k$ with respect to s_j yields

$$\begin{aligned} dz &= \alpha_j\, ds_j + \sum_k s_k\, d\alpha_k \\ &= \alpha_j\, ds_j + \sum_k s_k \alpha_k \left(\mathbb{I}[k = j] - \alpha_j \right) g_j^\top ds_j \\ &= \alpha_j\, ds_j + \alpha_j \left(s_j - \sum_k \alpha_k s_k \right) g_j^\top ds_j \\ &= \alpha_j \left[I + (s_j - z) g_j^\top \right] ds_j. \end{aligned} \tag{29}$$

Therefore,

$$\frac{\partial z}{\partial s_j} = \alpha_j \left[I + (s_j - z) g_j^\top \right],$$

which proves equation 28. □

For RMSNorm with learnable scale $D_\omega = \mathrm{diag}(\omega)$, the corresponding score gradient can be written explicitly as

$$\begin{aligned} R(s) &= D_\omega \frac{s}{\sqrt{\|s\|_2^2/d + \epsilon}}, \\ J_R(s) &= \frac{D_\omega}{r(s)} - \frac{(D_\omega s) s^\top}{d\, r(s)^3}, \qquad r(s) = \sqrt{\frac{\|s\|_2^2}{d} + \epsilon}, \\ g_j &= \frac{D_\omega q}{r(s_j)} - \frac{s_j s_j^\top D_\omega q}{d\, r(s_j)^3}. \end{aligned} \tag{30}$$

Suppose the routing correction remains stable and the direct depth representation receives non-vanishing attention:

$$\|s_j - z\|_2 \|g_j\|_2 \le \eta < 1, \qquad \alpha_j \ge \underline{\alpha} > 0. \tag{31}$$

Theorem 4 (Depth-independent cross-depth sensitivity)**.** *Under equation 31,*

$$\boxed{\sigma_{\min}\left(\frac{\partial z}{\partial s_j}\right) \ge \underline{\alpha}(1-\eta).} \tag{32}$$

The lower bound contains no multiplicative dependence on the depth distance between s_j and the target block.

Proof. From Lemma 2, define

$$u = s_j - z, \qquad v = g_j.$$

Then

$$\frac{\partial z}{\partial s_j} = \alpha_j (I + uv^\top).$$

For any vector x satisfying $\|x\|_2 = 1$, the reverse triangle inequality gives

$$\begin{aligned} \left\|(I + uv^\top)x\right\|_2 &\ge \|x\|_2 - \left\|uv^\top x\right\|_2 \\ &\ge 1 - \|u\|_2 \|v\|_2. \end{aligned} \tag{33}$$

Taking the minimum over all unit vectors yields

$$\sigma_{\min}(I + uv^\top) \ge 1 - \|u\|_2 \|v\|_2.$$

Using equation 31, $\|u\|_2\|v\|_2 \le \eta$ and $\alpha_j \ge \underline{\alpha}$. Therefore,

$$\begin{aligned} \sigma_{\min}\left(\frac{\partial z}{\partial s_j}\right) &= \alpha_j \sigma_{\min}(I + uv^\top) \\ &\ge \alpha_j (1 - \eta) \\ &\ge \underline{\alpha}(1 - \eta), \end{aligned} \tag{34}$$

which proves equation 32. □

By contrast, for purely adjacent propagation, if every intermediate block is locally contractive with $\|J_{\Phi_r}\|_2 \le \rho < 1$, then

$$J_{\mathrm{Adj}}^{j \to b} = J_{\Phi_{b-1}} \cdots J_{\Phi_j}, \qquad \left\| J_{\mathrm{Adj}}^{j \to b} \right\|_2 \le \rho^{b-j}. \tag{35}$$

Let $c_{\mathrm{BA}} = \underline{\alpha}(1 - \eta)$. Whenever

$$b - j > \frac{\log c_{\mathrm{BA}}}{\log \rho}, \tag{36}$$

the adjacent-path upper bound falls below the direct Block-Attention sensitivity bound. Thus, Block Attention introduces a cross-depth path that is not forced to undergo the exponential attenuation induced by repeated composition of contractive adjacent transformations.

G.3 LatentMoE Approximation–Computation Trade-off

We finally characterize LatentMoE. A conventional sparse MoE and its latent-space counterpart can be written as

$$F_{\mathrm{MoE}}(x) = \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) E_i(x),$$
$$F_{\mathrm{Latent}}(x) = W_{\mathrm{up}} \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) G_i(W_{\mathrm{down}} x), \tag{37}$$

where $\mathcal{S}_K(x)$ denotes the Top-K expert set. Because routing in TriScaleRec is computed from the original representation x, the two models can share the same routing distribution $\pi_i(x)$.

We assume that each active full-space expert admits a latent approximation

$$E_i(x) = W_{\mathrm{up}} G_i(W_{\mathrm{down}} x) + R_i(x), \qquad \|R_i(x)\|_2 \le \epsilon_{i,\ell}(x)\|x\|_2, \tag{38}$$

where $W_{\mathrm{down}} \in \mathbb{R}^{\ell \times d}$ and $W_{\mathrm{up}} \in \mathbb{R}^{d \times \ell}$.

Theorem 5 (Latent-space approximation of sparse experts)**.** *Under equation 38,*

$$\boxed{\|F_{\mathrm{MoE}}(x) - F_{\mathrm{Latent}}(x)\|_2 \le \epsilon_\ell(x)\|x\|_2, \qquad \epsilon_\ell(x) = \sum_{i \in \mathcal{S}_K(x)} \pi_i(x)\epsilon_{i,\ell}(x).} \tag{39}$$

Proof. Substituting equation 38 into the full-space sparse MoE gives

$$\begin{aligned} F_{\mathrm{MoE}}(x) &= \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) \left[W_{\mathrm{up}} G_i(W_{\mathrm{down}} x) + R_i(x) \right] \\ &= F_{\mathrm{Latent}}(x) + \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) R_i(x). \end{aligned} \tag{40}$$

Therefore,

$$\begin{aligned} \|F_{\mathrm{MoE}}(x) - F_{\mathrm{Latent}}(x)\|_2 &= \left\| \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) R_i(x) \right\|_2 \\ &\le \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) \|R_i(x)\|_2 \\ &\le \sum_{i \in \mathcal{S}_K(x)} \pi_i(x) \epsilon_{i,\ell}(x) \|x\|_2 \\ &= \epsilon_\ell(x)\|x\|_2. \end{aligned} \tag{41}$$

This proves equation 39. □

In particular, if $R_i(x) = 0$ for every active expert, then $F_{\mathrm{MoE}}(x) = F_{\mathrm{Latent}}(x)$. Thus, LatentMoE exactly recovers full-space experts whose transformations are representable through the shared latent space.

Let N denote the total number of experts, K the number of active experts, r the FFN expansion ratio, and c_{FFN} an architecture-dependent constant. The dominant total and active parameter costs satisfy

$$\begin{aligned} P_{\mathrm{total}}^{\mathrm{MoE}} &= c_{\mathrm{FFN}} N r d^2, & P_{\mathrm{active}}^{\mathrm{MoE}} &= c_{\mathrm{FFN}} K r d^2, \\ P_{\mathrm{total}}^{\mathrm{Latent}} &= 2d\ell + c_{\mathrm{FFN}} N r \ell^2, & P_{\mathrm{active}}^{\mathrm{Latent}} &= 2d\ell + c_{\mathrm{FFN}} K r \ell^2. \end{aligned} \tag{42}$$

Therefore, with fixed K and ℓ,

$$\frac{\partial P_{\mathrm{total}}^{\mathrm{Latent}}}{\partial N} = c_{\mathrm{FFN}} r \ell^2, \qquad \frac{\partial P_{\mathrm{active}}^{\mathrm{Latent}}}{\partial N} = 0, \tag{43}$$

showing that total expert capacity can grow with N without increasing the number of active expert parameters.

The corresponding active-parameter ratio is

$$\boxed{\frac{P_{\text{active}}^{\text{Latent}}}{P_{\text{active}}^{\text{MoE}}} = \frac{2}{c_{\text{FFN}} K r}\frac{\ell}{d} + \left(\frac{\ell}{d}\right)^2.} \tag{44}$$

When expert computation dominates the shared projection cost,

$$\frac{P_{\text{active}}^{\text{Latent}}}{P_{\text{active}}^{\text{MoE}}} \approx \left(\frac{\ell}{d}\right)^2, \qquad \ell \ll d. \tag{45}$$

Together, Theorem 3 shows that Block Attention retains conventional adjacent propagation as a limiting case, while Theorem 4 characterizes the additional direct cross-depth path that avoids mandatory multiplicative attenuation through every intermediate layer. Theorem 5 and equation 43–equation 45 further show how LatentMoE trades a controllable latent approximation for substantially reduced active expert computation.