Exploring Test-time Scaling via Prediction Merging on Large-Scale Recommendation

Fuyuan Lyu*
McGill University
Montreal, Canada
Mila – Quebec AI Institute
Montreal, Canada
fuyuan.lyu@mail.mcgill.ca

Zhentai Chen*
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China
zenithaitran@gmail.com

Jingyan Jiang
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China
jiangjingyan@sztu.edu.cn

Lingjie Li
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China
lilingjie@sztu.edu.cn

Xing Tang†
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China
xing.tang@hotmail.com

Xiuqiang He
School of Artificial Intelligence
Shenzhen Technology University
Shenzhen, China
he.xiuqiang@gmail.com

Xue Liu
McGill University
Montreal, Canada
Mila – Quebec AI Institute
Montreal, Canada
xueliu@cs.mcgill.ca

Abstract

Inspired by the success of language models (LM), scaling up deep learning recommendation systems (DLRS) has become a recent trend in the community. All previous methods tend to scale up the model parameters during training time. However, how to efficiently utilize and scale up computational resources during test time remains underexplored, which can prove to be a scaling-efficient approach and bring orthogonal improvements in LM domains. The key point in applying test-time scaling to DLRS lies in effectively generating diverse yet meaningful outputs for the same instance. We propose two ways: One is to explore the heterogeneity of different model architectures. The other is to utilize the randomness of model initialization under a homogeneous architecture. The evaluation is conducted across eight models, including both classic and SOTA models, on three benchmarks. Sufficient evidence proves the effectiveness of both solutions. We further prove that under the same inference budget, test-time scaling can outperform parameter scaling. Our test-time scaling can also be seamlessly accelerated with the increase in parallel servers when deployed online, without affecting the inference time on the user side. Code is available here[1].

*Equal Contribution
†Corresponding authors
[1]https://github.com/aTitye/TTS4CTR

SIGIR '26, Melbourne, VIC, Australia.

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3809740

CCS Concepts

• **Information systems → Data Mining**; • **Computing methodologies → Machine learning**.

Keywords

Scaling Laws, Test-time Scaling, Ranking, Recommendation

ACM Reference Format:
Fuyuan Lyu, Zhentai Chen, Jingyan Jiang, Lingjie Li, Xing Tang, Xiuqiang He, and Xue Liu. 2026. Exploring Test-time Scaling via Prediction Merging on Large-Scale Recommendation. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3805712.3809740

1 Introduction

Deep learning-based recommendation systems (DLRS) are deemed essential for multiple online services these days [27]. Modern DLRS takes multiple continuous dense features, such as date, and categorical sparse features, such as user clicked history. Categorical features are transformed into dense embedding representations throughout trainable embedding components. These dense embeddings are then fed into an interaction & prediction component, designed to explicitly capture the intricate interactions between features in an efficient manner [23, 28, 29].

Driven by the advancements in scaling law in the Language Model (LM) domain [15], scaling up the DLRS to fully leverage large volumes of online data remains an urgent need. Currently, the scaling on large-scale recommendation systems expands over two dimensions: (i) **width scaling**, which expands the number of interaction modules in a parallel manner. To avoid collisions and

achieve better representation, each module tends to correspond with a unique and independently trainable embedding table [21, 35]. Such scaling methods appear similar to Mixture-of-Expert [24, 31], but feature multiple embedding tables [11]. and (ii) **depth scaling**, which scales up the depth of the feature interaction module by designing a unified architecture that can be seamlessly stacked together [17, 47, 51]. However, both lines of work aim to scale up a single model by increasing the number of trainable parameters.

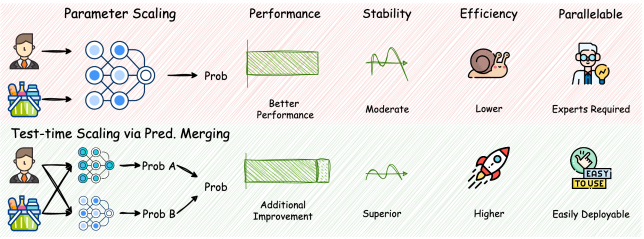


Figure 1: Comparison between Parameter Scaling and Test-time Scaling via Prediction Merging. The latter can benefit from additional improvement, superior stability, higher efficiency, and easy deployment.

Recently, the test-time scaling techniques have gathered increasing attention [26, 33]. The core idea of test-time scaling in the LM domain is to *deliberately increase computation at test time in the aim of increasing the quality of results* [33]. Test-time scaling has achieved **significant**, **resource efficient**, and **orthogonal** improvement over the parameter scaling solutions. Test-time scaling can also provide superior stability and ease of deployment in the DLRS domain, as elaborated in later Sections. Figure 1 visually explains the general idea and benefit of test-time scaling in DLRS. Yet, such an important idea has little exploration.

Although being a promising direction, the concept of test-time scaling cannot be effortlessly extended to DLRS. LMs can, by nature, generate diverse outputs, given the randomness in the factors such as the prompt or the decoding stage. However, for recommendations, especially those on a large scale, uniform input structures are preferred due to the requirement for large-scale online serving. Besides, the recommendation models tend to be trained under a unified data distribution, namely, all the features deemed useful by experts or algorithms [22, 41] are input into the model to achieve the best performance possible. Hence, how to obtain **diverse yet meaningful** outputs during test-time over large-scale recommendation models directly affects the utility of test-time scaling.

Here, we propose two approaches for obtaining diverse outputs from the same input. One approach is to leverage the heterogeneity among different recommendation models. Recent works have consistently proven the effectiveness of utilizing different recommendation architectures in capturing diverse interactions and correlations [35]. The other approach involves exploring randomness within a homogeneous architecture. Specifically, as initialization plays a crucial role in model training [9], we deliberately train models with different random seeds to encourage diversity in the logits spaces. Note that these two approaches can be combined. To efficiently fuse all the generated outputs, we propose to directly fuse them in the prediction space, namely, prediction merging.

We conduct extensive evaluations of eight recommendation models, both classic and SOTA, and their combinations. The evaluation is conducted across three benchmarks: Criteo, Avazu, and KDD12, with significant improvements observed in all cases. We further demonstrate the efficiency, the stability, and the orthogonality of test-time scaling. Importantly, as test-time scaling via prediction merging is by nature parallel, it can be easily deployed across different servers without a significant reduction in inference speed. Finally, we aim to investigation why test-time scaling works. Futher study reveals that diversity among predictions is a key factor in driving performance increases. Our contribution can be summarized as follows:

- We formally introduce an additional scaling dimension in DLRS: test-time scaling, which is orthogonal to the current parameter scaling approaches.
- We propose prediction merging, a model-agnostic framework that fuses predictions in the logit space, as the mechanism for test-time scaling. Under this framework, we investigate two sources of prediction diversity: heterogeneous architectures and homogeneous architectures with different random initializations. We further propose grouped training to improve scaling efficiency.
- Extensive evaluation over three large-scale benchmarks, eight recommendation models, and their combinations, validates the effectiveness, efficiency, and orthogonality of test-time scaling via prediction merging.

2 Methodology

2.1 Problem Definition

Large-scale Recommendation tends to formulate the problem as a binary classification. Specifically, it aims to predict the probability of a user interacting with given items. It can be defined as $\{\mathcal{X}, \mathcal{Y}\}$, with cardinality $|\{\mathcal{X}, \mathcal{Y}\}| = N$, where $\mathcal{X}$ is a multi-categorical feature set, including user, item, or contextual features, and $\mathcal{Y} \in \{0,1\}^N$ is a label set indicating whether the user interacted with the given item. Each feature x_i belongs to a feature field $\mathbf{x}_i$ with cardinality D_i, where D_i is the number of unique features in the feature field. Each feature field $\mathbf{x}_i$ is associated with an embedding table $\mathbf{E} \in \mathbb{R}^{D_i \times d}$, where d is the pre-defined embedding dimension size. This can be formulated as:

$$\mathbf{e}_i = \mathbf{E}(x_i). \tag{1}$$

All these embeddings are concatenated as $\mathbf{e} = [\mathbf{e}_1, \cdots, \mathbf{e}_n]$, where n indicates the number of feature fields. After obtaining all the embeddings, an interaction layer is applied to the embeddings, aiming to extract unique interactions. This can be formulated as:

$$\mathbf{h} = I(\mathbf{e}). \tag{2}$$

A prediction module is further concatenated to learn a mapping between the interaction space I and prediction label $\mathcal{Y}$, which can be formulated as: $\mathcal{F} : I \to \mathcal{Y}$. Binary cross-entropy (BCE) loss is commonly chosen as the objective function. For each instance $(x, y) \in \{\mathcal{X}, \mathcal{Y}\}$, the model can make the prediction as: $\hat{y} = \mathcal{F}(\mathbf{h})$. Hence, the final loss is formulated as follows:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} = -\frac{1}{N} \sum_{x,y \in \{\mathcal{X},\mathcal{Y}\}}^{N} y \log(\hat{y}) + (1-y)\log(1-\hat{y}). \tag{3}$$

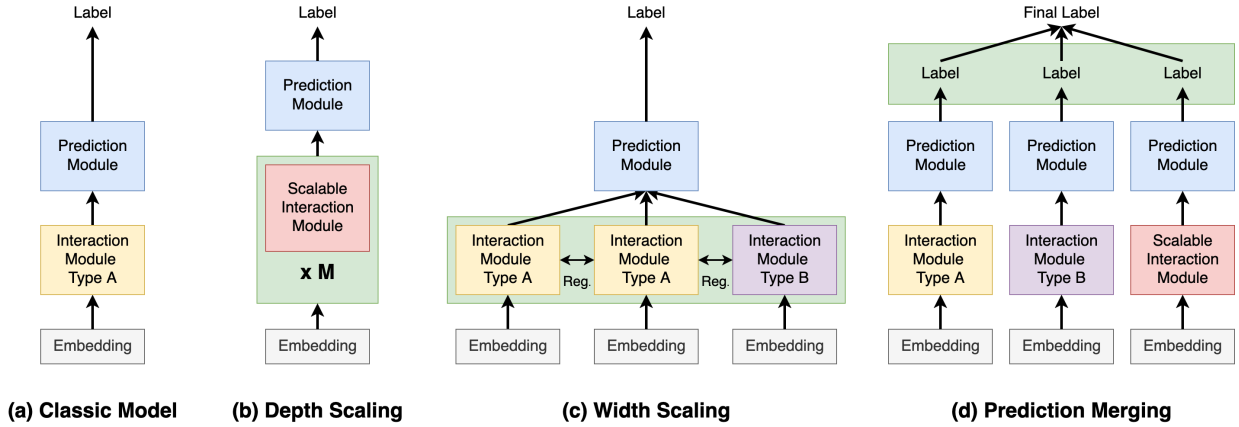


Figure 2: Visualization of different structures. (a) Classic Model design with Embedding, Interaction, and Prediction Module. (b) Depth Scabling Model design, which scales the depth of the interaction module and implicitly fuses information. (c) The Width Scaling Model design increases the number of interaction modules and the associated embedding table in parallel. It applies regularization among the interaction modules to encourage diversity and prevent overlapping. (d) Prediction Merging fuses the information directly in the logit space. For all structures, the interaction module and prediction module can have the same or different architectures. Green box indicates the information fusion process.

2.2 Parameter Scaling

In this section, we briefly categorize and formulate the two classes of parameter scaling.

2.2.1 Depth Scaling. Depth scaling [17, 47, 51] methods focus on increasing the depth of the interaction layer. One of the key bottlenecks lies in designing a scalable interaction layer. Hence, various layers, specifically tailored to be scalable and seamlessly concatenated, have been proposed. From a high-level perspective, we can formulate such a design as proposing a series of interaction layers $\{I^{(i)}|1 \leq i \leq M\}$, and each layer is stacked upon the former with the initial representation $\mathbf{h}^{(0)}$ being the embedding $\mathbf{e}$:

$$\mathbf{h}^{(i)} = I^{(i)}(\mathbf{h}^{(i-1)}),\ \mathbf{h}^{(0)} = \mathbf{e}. \tag{4}$$

Finally, the prediction module maps the final representation into the logits space, denoted as $\hat{y} = \mathcal{F}(\mathbf{h}^{(M)})$. The depth scaling models tend to be trained using similar objectives as Equation 3.

2.2.2 Width Scaling. Width Scaling [11, 21, 35] methods, on the other hand, intend to increase the width of the interaction layer. It is based on the hypothesis that various interaction layers may have different expertise in capturing users' interest. Hence, concatenating them may yield a better representation of users' interests. Due to the embedding collapse phenomenon [11], a trainable embedding table is associated with the corresponding interaction component for better scalability and capturing diverse behaviour patterns [24, 35]. Specifically, a model with M sets of embedding tables is defined as:

$$\begin{aligned} \mathbf{e}_i^{(m)} &= \mathbf{E}^{(m)}(x_i),\ 1 \leq m \leq M \\ \mathbf{e}^{(m)} &= [\mathbf{e}_1^{(m)}, \cdots, \mathbf{e}_n^{(m)}]. \end{aligned} \tag{5}$$

After obtaining the M sets of embedding $\{\mathbf{e}^{(m)} \mid 1 \leq m \leq M\}$, single or multiple experts, referring to different interaction modules, such as CIN [19], Cross Layer [36, 37], or Factorization Machine [10, 30], may be utilized to capture the diverse behaviour patterns. Specifically, each interaction expert $I^{(m)}$ is assigned to the m embedding $\mathbf{e}^{(m)}$ and aims to capture diverse interactions, formulated as:

$$\mathbf{h}^{(m)} = I^{(m)}(\mathbf{e}^{(m)}). \tag{6}$$

Here, each different interaction modules are averaged to obtain the final interactions

$$\mathbf{h} = \frac{1}{M}\sum_{m=1}^{M} h^{(m)}, \tag{7}$$

and the final prediction can be obtained as:$\hat{y} = \mathcal{F}(\mathbf{h})$. Additionally, to avoid the similarity among interaction representations, various diversity regularization terms [21, 35] have been proposed. This can be followed as:

$$\mathcal{L} = \mathcal{L}_{\text{pred}} + \lambda \mathcal{L}_{\text{reg}}(\{\mathbf{h}^{(m)} \mid 1 \leq m \leq M\}), \tag{8}$$

with λ indicating the strength of regularization and $\mathcal{L}_{\text{reg}}$ being the regularization term encouraging diversity among the interaction set $\{\mathbf{h}^{(m)} \mid 1 \leq m \leq M\}$.

2.3 Test-time Scaling via Prediction Merging

The major difference between existing parameter scaling methods and prediction merging lies in two aspects. First, prediction merging encourages the generation of diverse outputs directly, whereas parameter scaling methods only generate a single prediction. Unlike existing models [11, 21, 35, 47], which operate on the interaction layer, prediction merging directly operates on the prediction logit space $\mathcal{Y}$ without making assumptions about the architecture of the individual model. Second, the number of supervision signals for prediction merging is higher than that for parameter scaling. Prediction merging involves training models $\mathcal{G}^{(m)} : x \to y$ independently or in a grouped manner, whereas existing solutions train all experts jointly with only one supervision signal, incorporating regularization based on human-prior knowledge.

Hence, under the prediction merging framework, multiple models, regardless of their architectures or prior assumptions, can be integrated seamlessly. Specifically, given a family of models $\{\mathcal{F}^{(m)} \mid 1 \leq m \leq M\}$. Hence, for each data instance $(x, y) \in \{\mathcal{X}, \mathcal{Y}\}$, model $\mathcal{F}^{(m)}$ make its prediction as

$$\hat{y}^{(m)} = \mathcal{G}^{(m)}(x). \tag{9}$$

After obtaining the prediction sets $\{\hat{y}^{(m)} \mid 1 \leq m \leq M\}$, prediction merging directly averages all predictions and yields the final prediction as:

$$\tilde{y} = \frac{1}{M} \sum_{m=1}^{M} \hat{y}^{(m)}. \tag{10}$$

Algorithm 1 describe the detailed solution in obtaining the model set $\{\mathcal{G}^{(m)} \mid 1 \leq m \leq M\}$.

Algorithm 1 The Training Phase for Prediction Merging

Require: training dataset $\{\mathcal{X}, \mathcal{Y}\}$, initial model set $\{\mathcal{G}_0^{(m)} \mid 1 \leq m \leq M\}$
Ensure: model set $\{\mathcal{G}^{(m)} \mid 1 \leq m \leq M\}$
1: Split initial model set into K mutually excluded subset $\mathcal{S}_1, \cdots, \mathcal{S}_K$, with each containing at least one element.
2: k = 0
3: **while** k < K **do**
4: k = k + 1
5: **while** not converged **do**
6: Sample a mini-batch from the training dataset $\{\mathcal{X}, \mathcal{Y}\}$
7: Calculate the prediction $\tilde{y}_k$ with Equation 10 given $\mathcal{S}_k$
8: Tune models in $\mathcal{S}_k$ given the objective in Equation 3
9: **end while**
10: **end while**

2.3.1 Grouped Training Strategies. To achieve higher scaling efficiency and enhance the collaborative performance of the model set, we introduce *grouped training*. Unlike independent training where each model is optimized in isolation and prediction merging is only applied during test-time, grouped training merge several models into groups and apply the prediction merging before training. Specificially, it concatenates multiple models into a single functional unit during the forward and backward passes. This can be formulated as:

$$\hat{y}_g^{(i)} = \frac{1}{M_i} \sum_{m \in M_i} \hat{y}^{(m)}, \quad \hat{y}^{(m)} = \mathcal{G}^{(m)}(x). \tag{11}$$

Here $\hat{y}_g^{(i)}$ is optimized to approximate the ground truth y, meaning that models in group M_i are trained jointly as if they are one model.

The key distinction from independent training lies in the loss computation. In independent training, each model minimizes its own loss $\mathcal{L}(\hat{y}^{(m)}, y)$ separately. In grouped training, the loss is computed on the **group-level averaged prediction**: $\mathcal{L}_g = \mathcal{L}(\hat{y}_g^{(i)}, y)$. During backpropagation, the gradient received by each model m in group M_i is:

$$\frac{\partial \mathcal{L}_g}{\partial \theta^{(m)}} = \frac{1}{|M_i|} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}_g^{(i)}} \cdot \frac{\partial \hat{y}^{(m)}}{\partial \theta^{(m)}} \tag{12}$$

Since $\hat{y}_g^{(i)}$ depends on all models in the group, each model's gradient is implicitly conditioned on its group members' predictions. This creates an implicit regularization effect: if other models in the group already predict correctly for a sample, the gradient signal for that sample is reduced, encouraging each model to focus on complementary error patterns rather than redundantly learning the same mapping.

After each group training is done, the final prediction is average across the group prediction with the group cardinality as weights, denoted as:

$$\tilde{y} = \frac{1}{M} \sum_{i=1}^{K} \|M_i\| * \hat{y}_g^{(i)}, \quad \sum_{i=1}^{K} \|M_i\| = M \tag{13}$$

3 Experiment

In this section, we aim to evaluate and answer the following research questions:

- **RQ1**: Is test-time scaling (TTS) effective for recommendation?
- **RQ2**: How does TTS compare to parameter scaling in terms of performance and efficiency?
- **RQ3**: Is TTS orthogonal to parameter scaling?
- **RQ4**: How does prediction merging compare to model merging?
- **RQ5**: Why can TTS be achieved via prediction merging, and does training-based TTS outperform training-free version?

3.1 Experimental Setup

3.1.1 Datasets. We conduct our experiments on three public real-world datasets: **Criteo** [14], **Avazu** [38], and **KDD12** [1]. We follow the data preprocessing and dataset splitting protocols from previous work [22].

3.1.2 Metrics. Following the previous works [10, 30], we use the common evaluation metrics for CTR prediction: **AUC** (Area Under ROC) and **Log loss** (cross-entropy). Note that **0.1**% improvement in AUC is considered significant [10, 28].

3.1.3 Backbones and Baselines. We evaluate the test-time scaling and other scaling techniques on two types of backbone models: (i) classical DLRS models: FNN [50], IPNN [28], DeepFM [10], DCN [36], DCNv2 [37], and FinalMLP [25], and (ii) depth scaling models: Wukong [47] and RankMixer [51]. We further compare with two width scaling techniques orthogonal to backbone models, namely MultiEmbed (ME) [11], D-MoE [35].

3.1.4 Group training setup. Here we report the group training setup utilized in Table 3. Due to the space limit, here we report only part of the result. The rest of them is contained in the repository.

3.2 RQ1: Main Results

In this section, we evaluate the scalability of prediction merging in two situations: (i) homogenous architectures and (ii) heterogeneous architectures.

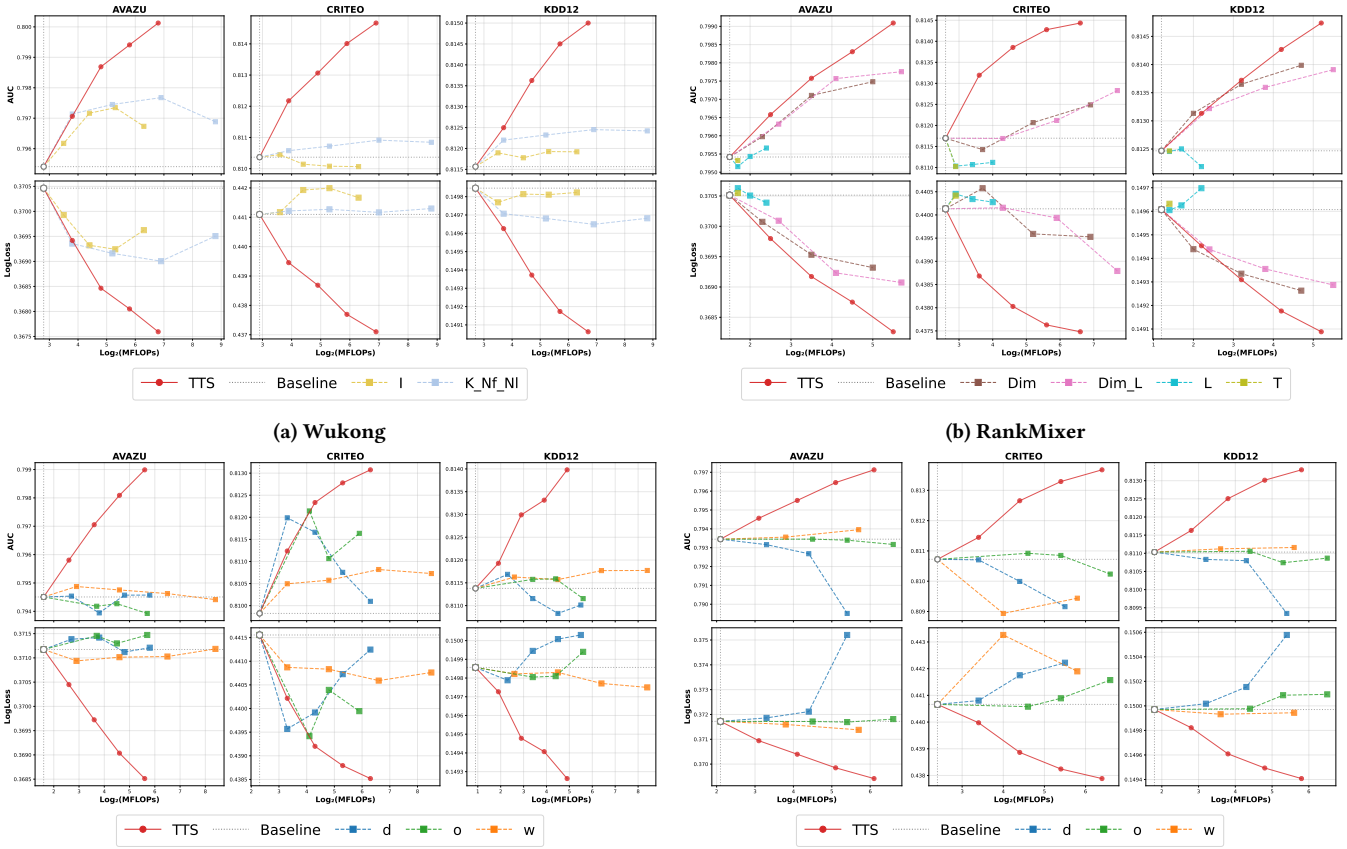
3.2.1 Comparison against Depth Scaling.

3.2.2 Homogenous Architectures. For homogeneous architectures, we scale up the same architecture via initialization randomness, achieving up to 16× scaling. Based on Table 1, we can make the following observations. Firstly, we can observe that the performance,

Table 1: Scalability on Homogenous Architecture.

	Method	FNN		IPNN		DeepFM		DCN		DCNv2		FinalMLP		Wukong		RankMixer	
		AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓
Avazu	Backbone	0.7888	0.3761	0.7911	0.3744	0.7869	0.3757	0.7889	0.3753	0.7945	0.3712	0.7935	0.3717	0.7954	0.3705	0.7954	0.3705
	2×	0.7897*	0.3755	0.7926*	0.3733	0.7878	0.3769	0.7898*	0.3753	0.7958*	0.3704	0.7946*	0.3709	0.7971*	0.3695	0.7966*	0.3698
	4×	0.7903*	0.3744	0.7935*	0.3726	0.7911	0.3733	0.7905*	0.3747	0.7971*	0.3697	0.7955*	0.3704	0.7987*	0.3685	0.7976*	0.3692
	8×	0.7909*	0.3740	0.7938*	0.3724	0.7923	0.3727	0.7907*	0.3745	0.7981*	0.3691	0.7965*	0.3699	0.7994*	0.3681	0.7983*	0.3688
	16×	0.7913*	0.3737	0.7941*	0.3722	0.7930	0.3730	0.7910*	0.3739	0.7990*	0.3685	0.7971*	0.3694	0.8001*	0.3676	0.7991*	0.3683
Criteo	Backbone	0.8103	0.4411	0.8110	0.4404	0.8085	0.4428	0.8104	0.4412	0.8098	0.4416	0.8107	0.4407	0.8104	0.4411	0.8117	0.4401
	2×	0.8110	0.4405	0.8118	0.4397	0.8102	0.4413	0.8110	0.4405	0.8112	0.4402	0.8114	0.4400	0.8122*	0.4395	0.8132*	0.4387
	4×	0.8116*	0.4400	0.8122*	0.4393	0.8110*	0.4406	0.8116*	0.4400	0.8123*	0.4392	0.8127*	0.4389	0.8131*	0.4387	0.8139*	0.4380
	8×	0.8121*	0.4396	0.8125*	0.4392	0.8115*	0.4401	0.8122*	0.4394	0.8138*	0.4388	0.8133*	0.4382	0.8140*	0.4377	0.8143*	0.4376
	16×	0.8124*	0.4331	0.8130*	0.4387	0.8117*	0.4400	0.8124*	0.4392	0.8131*	0.4385	0.8137*	0.4378	0.8147*	0.4371	0.8144*	0.4375
KDD12	Backbone	0.8108	0.1500	0.8118	0.1497	0.8085	0.1505	0.8108	0.1500	0.8114	0.1499	0.8110	0.1500	0.8116	0.1498	0.8125	0.1496
	2×	0.8114	0.1498	0.8126*	0.1496	0.8099*	0.1502	0.8113	0.1499	0.8119	0.1497	0.8116	0.1498	0.8125*	0.1496	0.8131*	0.1495
	4×	0.8122	0.1497	0.8135*	0.1493	0.8109*	0.1500	0.8120*	0.1497	0.8130*	0.1494	0.8125*	0.1496	0.8136*	0.1494	0.8137*	0.1493
	8×	0.8127*	0.1496	0.8141*	0.1492	0.8115*	0.1498	0.8127*	0.1496	0.8133*	0.1494	0.8130*	0.1495	0.8145*	0.1492	0.8143*	0.1492
	16×	0.8132*	0.1494	0.8144*	0.1491	0.8119*	0.1497	0.8131*	0.1495	0.8140*	0.1493	0.8132*	0.1494	0.8150*	0.1491	0.8147*	0.1491

Here ∗ denotes statistically significant improvement (measured by a two-sided t-test with p-value < 0.05) over the backbone in terms of AUC.



(a) Wukong (b) RankMixer

(c) DCNv2 (d) FinalMLP

Figure 3: Performance scaling curves across three datasets (Avazu, Criteo, and KDD12) as a function of multiplicative FLOPs scaling factor. Each subfigure shows a model's performance under different computational budgets. Scalable architectures (Wukong and RankMixer) exhibit consistent performance improvements following scaling laws, while others (DCNv2 and FinalMLP) demonstrate diminishing or negative returns at higher FLOPs, indicating limited parameter scalability.

as measured by both AUC and Logloss, consistently improves as the number of experts scales. Significant improvements are observed in most cases. Secondly, on certain models, significant improvements can be observed when merging merely two models, hinting at the efficiency of test-time scaling on homogeneous architectures. Finally, we can observe that depth scaling architectures, namely Wukog and Rankmixer, exhibit better scalability than others.

3.2.3 Heterogenous Architectures. For heterogeneous architectures, we investigate three combinations: (1) Wukong + Rankmixer, (2) Rankmixer + DCNv2, and (3) Wukong + DCNv2. Results are shown

Table 2: Scalability on Heterogeneous Architecture.

	Method	Wukong+Rankmixer		Rankmixer+DCNv2		Wukong+DCNv2	
		AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓
Avazu	1×	0.7977	0.3690	0.7964	0.3698	0.7971	0.3694
	2×	0.7986*	0.3685	0.7968	0.3696	0.7977	0.3690
	4×	0.8003*	0.3676	0.7971	0.3694	0.7987*	0.3684
	8×	0.8017*	0.3667	0.7972	0.3693	0.7993*	0.3680
	16×	0.8023*	0.3663	0.7983*	0.3689	0.8000*	0.3676
	32×	0.8028*	0.3661	0.7987*	0.3689	0.8004*	0.3674
Criteo	1×	0.8138	0.4380	0.8136	0.4382	0.8132	0.4384
	2×	0.8145*	0.4373	0.8141*	0.4376	0.8139	0.4378
	4×	0.8152*	0.4367	0.8145*	0.4373	0.8146*	0.4372
	8×	0.8152*	0.4367	0.8147*	0.4370	0.8147*	0.4371
	16×	0.8154*	0.4365	0.8148*	0.4370	0.8150*	0.4368
	32×	0.8155*	0.4364	0.8149*	0.4369	0.8150*	0.4368
KDD12	1×	0.8132	0.1495	0.8130	0.1495	0.8128	0.1495
	2×	0.8135	0.1494	0.8134	0.1494	0.8134	0.1494
	4×	0.8137*	0.1493	0.8137	0.1493	0.8137	0.1493
	8×	0.8143*	0.1498	0.8140	0.1494	0.8143	0.1492
	16×	0.8144*	0.1502	0.8144	0.1493	0.8148*	0.1491
	32×	0.8153*	0.1498	0.8149	0.1493	0.8155*	0.1489

* denotes significant improvement between the current scaling and the base model on AUC, measured by a two-tailed t-test with p-value smaller than 0.05.

Table 3: Group Training Configurations.

Dataset	Model	Scale of Group M/K				
		1x	2x	4x	8x	16x
Avazu	DCNv2	1	2	4	8	8
	RankMixer	1	2	4	8	16
	Wukong	1	2	2	4	4
	W+R	1	2	4	4	8
Criteo	DCNv2	1	2	2	2	2
	RankMixer	1	1	1	1	1
	Wukong	1	1	1	2	2
	W+R	1	1	1	1	1
KDD12	DCNv2	1	1	2	2	16
	RankMixer	1	2	2	4	8
	Wukong	1	1	2	4	4
	W+R	1	1	1	2	4

in Table 2. We make the following observations. First, similar to homogeneous architectures, prediction merging exhibits consistent scalability in heterogeneous settings. Performance improves as the number of models increases across all combinations. Second, combining strong architectures generally yields better performance. Wukong + Rankmixer achieves the optimal results in most cases. This aligns with the previous observation that Wukong and Rankmixer demonstrate superior scalability.

3.3 RQ2: Comparison against Scaling Baselines

In this section, we compare test-time scaling against two parameter scaling paradigms: depth scaling (increasing model depth or capacity) and width scaling (multiple embedding or interaction modules), and analyze their efficiency trade-offs.

We demonstrate that under equivalent computational budgets (FLOPs), test-time scaling via prediction merging achieves stronger performance than parameter scaling (Figure 3), while also enabling lower inference latency through parallel serving—though the latency advantage requires sufficient parallel resources. This is an observation witnessed previously in the language model domain [33]. We visualize both scaling solutions as performance-efficiency curves in Figure 3, evaluating depth scaling models (Wukong and Rankmixer) and classic models (DCNv2 and FinalMLP). We make the following observations: First, test-time scaling generally outperforms parameter scaling across all datasets and backbones, showcasing our solution's superiority. Second, parameter scaling performance varies significantly across datasets, with certain models demonstrating good scalability on some datasets but limited scalability on others. Finally, depth scaling incurs a two-fold cost: exploring various scaling dimensions (e.g., embedding dimension, depth, or both) and extensive hyperparameter tuning for each configuration.

3.3.1 Comparison against Width Scaling. We further compare prediction merging (PM) against width scaling methods: MultiEmbed (ME) [11] and D-MoE [35]. Unlike PM's late fusion at the logits level, these methods fuse information at the embedding or interaction stage. Results are shown in Table 4.

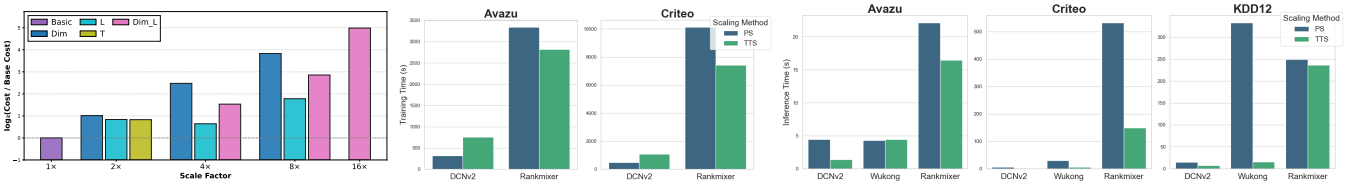
Table 4: Comparison against Width Scaling

	Model	ME		D-MoE		PM	
		AUC↑	Logloss↓	AUC↑	Logloss↓	AUC↑	Logloss↓
Avazu	W*2	0.7969	0.3697	**0.7972**	0.3709	0.7970	**0.3695**
	R*2	0.7949	0.3708	0.7954	0.3710	**0.7966***	**0.3698***
	R+W	0.7953	0.3707	0.7976	0.3722	**0.7977**	**0.3690***
Criteo	W*2	0.8106	0.4410	0.8111	0.4408	**0.8122***	**0.4395***
	R*2	0.8115	0.4406	0.8110	0.4406	**0.8132***	**0.4387***
	R+W	0.8113	0.4405	0.8110	0.4410	**0.8138***	**0.4380***
KDD12	W*2	0.8125	0.1497	**0.8130**	0.1497	0.8125	**0.1496**
	R*2	0.8126	0.1497	0.8128	0.1496	**0.8131**	**0.1495**
	R+W	0.8119	0.1499	**0.8132**	0.1496	0.8131	**0.1495**

Here, R and W are abbreviations of Rankmixer and Wukong, respectively. The best and second-best performed structure is highlighted in **bold** and underline font, respectively. * denotes significant improvement between the best and second-best performed baselines, measured by a two-tailed t-test with p-value smaller than 0.05.

3.3.2 Efficiency Analysis. We compare efficiency between parameter scaling (8×) and test-time scaling (8×). All experiments are conducted on an 8-core Intel Xeon Gold 6459C CPU with one A100 GPU (48GB). Figure 4 presents three efficiency dimensions: parameter tuning, training as well as inference.

Tuning Cost (Figure 4a): Compared with test-time scaling, parameter scaling requires extensive hyperparameter search across scaling dimensions. We can easily observe that the tuning costs increases substantially at higher scaling factors. Such a computation overhead that becomes prohibitive in industrial settings. Comparable, test-time scaling can be achieved throughout training several smaller models, usually with identitical setup. This would result in reduction of tuning cost. **Training & Inference Time** (Figures 4b, 4c): We can observe that test-time scaling yields superior inference speed on scalable models (Wukong, RankMixer), as the parallel nature avoids sequential bottlenecks inherent in deeper models. As for the training efficiency, test-time scaling yields stronger efficiency for Rankmixer while taking more time on DCNv2. This is because for simpler architectures like DCNv2, I/O bandwidth becomes the bottleneck when running multiple processes, leading to lower efficiency of test-time scaling in terms of training.



(a) Hyperparameter Tuning Cost (b) Training Time per Epoch (c) Inference Time on Validation

Figure 4: Efficiency comparison between parameter scaling and test-time scaling: (a) hyperparameter tuning overhead increases substantially with scaling factor for depth scaling; (b) training time per epoch; (c) inference latency—test-time scaling yields superior speed on scalable models (Wukong, RankMixer) due to parallelization.

3.4 RQ3: Orthogonality Experiment

We investigate whether parameter scaling is orthogonal to test-time scaling. We scale up both Wukong and Rankmixer via the best-performing dimension by up to 16×, and further apply test-time scaling by 16×. Based on Table 5, we can make the following observations.

First, parameter scaling demonstrates consistent but saturating improvements across all configurations. Models exhibit diminishing returns as parameters increase, with the plateau effect particularly pronounced on certain datasets. These observations highlight the fundamental limitation of parameter scaling when training data is fixed—additional model capacity cannot be effectively utilized without proportional data growth.

Second, compared with the base model, test-time scaling shows consistent improvement in both AUC and Logloss with scaling up. We observe that the performance gap between the test-time scaled solution and the base model remains positive across all parameter scaling levels, though its magnitude decreases as the base model grows larger. We attribute this to the saturation effect of parameter scaling under fixed training data: as model capacity increases beyond what the data can support, the model begins to overfit noise, leaving less room for variance reduction via prediction merging. Importantly, this diminishing return reflects a data-capacity mismatch rather than a coupling between the two scaling dimensions—the improvement mechanisms remain orthogonal in nature. A similar phenomenon is also observed on the LMs [4, 33].

3.5 RQ4: Comparison against Model Merging

Model merging has been an effective trick in language model domains. Specifically speaking, it fuses multiple foundation models with individually finetuned on different tasks, namely expert models [3, 13, 42, 45]. After fusion, these merged models can yield comparable performance on individual tasks compared with the corresponding expert models. In the meantime, the number of models and inference cost can be greatly reduced [3].

In this section, we elaborate on how such model merging techniques can be combined with large-scale recommendation models, which, different from LMs, are trained independently from scratch. Here, we adopt one of the commonly used techniques, which directly averages the weights among different models [45]. We report the performance of both prediction merging and model merging in Table 6. Observations are listed as follows.

First, prediction merging consistently outperforms model merging across all setups. This highlights the superiority of prediction merging over model merging in large-scale recommendation tasks. Second, unlike prediction merging, where performance increases as the number of models increases, model merging exhibits a consistent decrease in performance in nearly all setups when the number of models is increased. As the number of models increases to 16×, the performance of model merging may decrease to around 0.5 in terms of AUC, which renders the final results indistinguishable from random guessing. Such a surprising result highlights that model merging techniques may not be easily adopted in large-scale recommendation tasks. We believe this is likely due to the fact that in the language model domain, the merged model tends to share a common ancestor, a pre-trained foundation model that was not tuned on specific tasks. Hence, these parameter distribution may not differs too much from each other. However, in our setups, different experts are trained independently with individually random initialization. Hence, directly merging them can be hard without tailored techniques.

3.6 RQ5: Case Study

In this section, we investigate why test-time scaling works and whether training-based approaches outperform training-free alternatives.

3.6.1 Why Prediction Merging Works. In this section, we aim to further explore why prediction merging work. Specificially, we explore two distinct configurations for test-time scaling under fixed computational budgets. We denote them as **Basic** and **Group** $K\times$ strategies:

- **Basic:** Each model is trained independently with different random seeds, and their predictions are aggregated at inference time following Equation 10.
- **Group** $K\times$**:** Models are equally partitioned into groups of size K, where each group is trained jointly following Section 2.3.1.

Separation and Sharpness Analysis. To understand the mechanism behind test-time scaling, we visualize the prediction distribution characteristics in Figure 5. Two key metrics are examined:

- *Separation* [6]: the standardized distance between positive and negative class predictions, defined as:

$$\text{Separation} = \frac{\mu_+ - \mu_-}{\sqrt{\sigma_+^2 + \sigma_-^2}} \tag{14}$$

Table 5: Performance comparison of Wukong and Rankmixer with different scaling factors on Avazu and KDD12 datasets. The values in parentheses indicate the improvement over the corresponding Base model.

Dataset	Model	Scaling Type	Param 1x		Param 2x		Param 4x		Param 8x		Param 16x	
			AUC↑	LogLoss↓	AUC↑	LogLoss↓	AUC↑	LogLoss↓	AUC↑	LogLoss↓	AUC↑	LogLoss↓
Avazu	Wukong	Base	0.7954	0.3705	0.7969	0.3695	0.7974	0.3692	0.7972	0.3693	0.7972	0.3693
		Test-time 16x	0.7984 (+30)	0.3686 (-19)	0.7994 (+25)	0.3680 (-15)	0.7995 (+21)	0.3679 (-13)	0.7995 (+23)	0.3679 (-14)	0.7995 (+23)	0.3679 (-14)
	Rankmixer	Base	0.7954	0.3705	0.7962	0.3700	0.7975	0.3693	0.7976	0.3691	0.7979	0.3690
		Test-time 16x	0.7968 (+14)	0.3696 (-9)	0.7973 (+11)	0.3692 (-8)	0.7991 (+16)	0.3682 (-11)	0.7993 (+17)	0.3681 (-10)	0.7988 (+9)	0.3684 (-6)
Criteo	Wukong	Base	0.8104	0.4411	0.8106	0.4410	0.8109	0.4408	0.8108	0.4407	0.8110	0.4406
		Test-time 16x	0.8138 (+34)	0.4381 (-30)	0.8124 (+18)	0.4394 (-16)	0.8126 (+17)	0.4392 (-16)	0.8125 (+17)	0.4392 (-15)	0.8125 (+15)	0.4392 (-14)
	Rankmixer	Base	0.8117	0.4401	0.8118	0.4397	0.8123	0.4394	0.8128	0.4390	0.8128	0.4388
		Test-time 16x	0.8144 (+27)	0.4375 (-26)	0.8129 (+11)	0.4386 (-11)	0.8129 (+6)	0.4389 (-5)	0.8133 (+5)	0.4385 (-5)	0.8134 (+6)	0.4382 (-6)
KDD12	Wukong	Base	0.8116	0.1498	0.8120	0.1497	0.8119	0.1497	0.8122	0.1497	0.8121	0.1497
		Test-time 16x	0.8135 (+19)	0.1494 (-4)	0.8135 (+15)	0.1494 (-3)	0.8137 (+18)	0.1494 (-3)	0.8139 (+17)	0.1493 (-4)	0.8139 (+18)	0.1493 (-4)
	Rankmixer	Base	0.8125	0.1496	0.8130	0.1495	0.8137	0.1493	0.8138	0.1493	0.8140	0.1493
		Test-time 16x	0.8136 (+11)	0.1493 (-3)	0.8141 (+11)	0.1492 (-3)	0.8147 (+10)	0.1491 (-2)	0.8144 (+6)	0.1491 (-2)	0.8152 (+12)	0.1490 (-3)

Wukong scaled the dimension K_Nf_Nl from 1× to 16×. **Rankmixer** scaled Dim_L at 16×, and Dim otherwise.

Table 6: Model vs. Prediction Merging.

	Method	DCNv2		FinalMLP		Wukong		RankMixer	
		AUC↑	LogLoss↓	AUC↑	LogLoss↓	AUC↑	LogLoss↓	AUC↑	LogLoss↓
Avazu	Backbone-Model	0.7945	0.3711	0.7869	0.3757	0.7954	0.3705	0.7954	0.3705
	2×-Model	0.7692	0.5279	0.7076	0.5293	0.7209	0.6680	0.7697	0.6035
	2×-Pred	0.7958	0.3704	0.7946	0.3709	0.7971	0.3695	0.7966	0.3698
	4×-Model	0.7445	0.6335	0.5652	0.5337	0.6068	0.6923	0.7402	0.6739
	4×-Pred	0.7971	0.3697	0.7955	0.3704	0.7987	0.3685	0.7976	0.3792
	16×-Model	0.7412	0.6830	0.5000	0.5384	0.4856	0.6846	0.5725	0.6845
	16×-Pred	0.7990	0.3685	0.7971	0.3694	0.8001	0.3676	0.7991	0.3683
Criteo	Backbone-Model	0.8098	0.4416	0.8107	0.4407	0.8104	0.4411	0.8117	0.4401
	2×-Model	0.7694	0.5515	0.7923	0.5900	0.7601	0.6605	0.7809	0.6253
	2×-Pred	0.8112	0.4402	0.8114	0.4400	0.8122	0.4395	0.8132	0.4387
	4×-Model	0.7577	0.6271	0.7665	0.6331	0.6997	0.6811	0.7221	0.6531
	4×-Pred	0.8123	0.4392	0.8127	0.4389	0.8131	0.4387	0.8139	0.4380
	16×-Model	0.7458	0.6783	0.5027	0.6397	0.5924	0.6830	0.6005	0.6562
	16×-Pred	0.8131	0.4385	0.8137	0.4378	0.8147	0.4371	0.8144	0.4375
KDD12	Backbone-Model	0.8114	0.1499	0.8110	0.1500	0.8116	0.1498	0.8125	0.1496
	2×-Model	0.7704	0.3908	0.7130	0.4670	0.6683	0.6402	0.7538	0.5589
	2×-Pred	0.8119	0.1497	0.8116	0.1498	0.8125	0.1496	0.8131	0.1495
	4×-Model	0.6596	0.5843	0.4812	0.4813	0.5407	0.6574	0.6943	0.6321
	4×-Pred	0.8130	0.1494	0.8125	0.1496	0.8136	0.1494	0.8137	0.1493
	16×-Model	0.6868	0.6589	0.500	0.4802	0.5038	0.6537	0.5091	0.6448
	16×-Pred	0.8140	0.1493	0.8132	0.1494	0.8150	0.1491	0.8147	0.1491

Model denotes model weight merging (averaging parameters from models), while **Pred** denotes prediction merging (averaging outputs).

where μ_+, μ_- and σ_+, σ_- denote the means and standard deviations of positive and negative class predictions, respectively.

- *Sharpness*: the inverse of prediction variance, measuring how concentrated the predictions are:

$$\text{Sharpness} = \frac{1}{2}\left(\frac{1}{\sigma_+} + \frac{1}{\sigma_-}\right) \tag{15}$$

Based on the result shown in Figure 5, we can observe two distinct trends. First, the box plots reveal the distribution of separation. We can easily observe that as the equivalent scale increases, the *separation* between positive and negative predictions consistently grows. Together with the consistent performance improvement during scaling, this indicates that the AUC improvement is positively correlated with better class separation. Second, examining the scatter plots with dashed outlines (where group scale equals equivalent scale, i.e., single group models), we observe that sharpness decreases as scale increases. More interestingly, for each group scale, the colored points form an upward-converging triangular pattern as the total scale increases. This reveals two complementary mechanisms:

- **Basic aggregation** reduces prediction variance by averaging multiple predictions, effectively compressing the positive and negative distributions into narrower, sharper peaks.
- **Group training** enhances individual model capacity with increased variance, the separation grows substantially, pushing positive and negative distributions further apart.

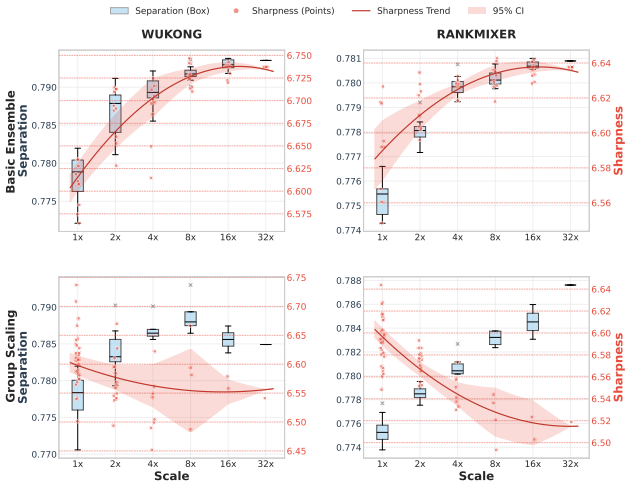


Figure 5: Separation and sharpness analysis across different scaling configurations on Avazu. Box plots show increasing separation with scale. Scatter plots reveal the complementary mechanisms between basic and group strategies.

Diversity and Prediction Selection Strategies. In this paragraph, we aim to further investigate why basic aggregation can sharpe the prediction distribution. Specifically, we investigate the diversity among predictions, separations and their effect on the final AUC. We plot the relationship figures in Figure 6 where two positive correlation with high correlation (r > 0.98) can be observed. This shows that the diversity among more predictions directly contributes to the increase of separation, which further leads to the improvement of performance.

We also compare different prediction selection strategies for merging Figure 6. *Greedy* exhaustively searches for the optimal subset

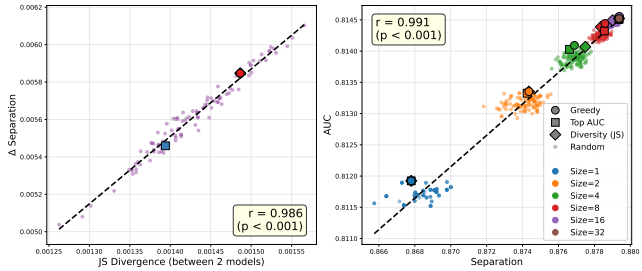


Figure 6: Correlation between Separation improvement, JS divergence and AUC on Criteo with Rankmixer. JS-divergence is computed via randomly selecting two models among all.

maximizing AUC; *Top AUC* selects models with highest individual AUC; *Diversity (JS)* selects models with highest pairwise JS divergence; *Random* samples models uniformly. The Diversity (JS) strategy performs comparably to Greedy for sizes up to 8, outperforming both Top AUC and Random selections. This corroborates our findings that prediction diversity is a key driver of ensemble performance.

Connection to Bias-Variance Tradeoff. Test-time scaling can be understood through the lens of bias-variance tradeoff. Parameter scaling increases model capacity but risks overfitting when training data is limited (high variance). In contrast, test-time scaling via prediction merging reduces variance by averaging diverse predictions while maintaining low bias with each constituent model capturing different aspects of the data distribution. This provides a principled approach to improve generalization without the overfitting risks associated with simply enlarging model parameters. Under limited scaling budget, such a trade-off can be achieved throughout the group training design.

3.6.2 Training-Based vs. Training-Free Approaches. As described in previous sections, our test-time scaling approach requires training multiple models. Therefore, a natural question arises: **can test-time scaling be achieved through training-free methods that generate multiple predictions from a single model trained once?** In this section, we . Specificially, we investigate two common and popular approaches from previous literature and test if these approachs works under our settings.

- *Monte Carlo Dropout (MC-Dropout)* [18] performs Bayesian uncertainty approximation by running multiple stochastic forward passes with different dropout masks during inference. Each dropout configuration samples from an approximate posterior, theoretically enabling ensemble-like behavior without training multiple models. We tested various dropout rates and found that dropout rate = 0.1 yields minimal performance degradation.
- *Feature Masking (FM)* [7] applies frequency-aware masking during inference. Feature occurrence frequencies from training data serve as sampling probabilities, where frequent features are retained while rare features are stochastically masked. Multiple masked predictions are averaged to reduce sensitivity to noisy low-frequency features. We set mask_top_frac = 0.9 (retaining top 90% frequent features) with 8 mask paths.

Pairwise Prediction Analysis. Figure 7 presents a 5×5 matrix comparing five prediction sources: Naive (baseline), Naive+MC (MC-Dropout), Naive+FM (Feature Masking), DP (trained with dropout), and DP+MC. The diagonal in Figure 7 shows that all models exhibit similar individual overlap areas, indicating comparable separation capabilities. Critically, the upper triangle reveals that training-free variants show near-identical predictions to their base models with high correlation, while cross-training, namely Naive VS. DP, exhibits meaningful diversity. Such an observation highlights that the diversity of prediction can only be achieved throughout the training stage. The lower triangle confirms that ensembles of similar predictions yield no improvement, whereas cross-training combinations achieve non-neglectable overlap reduction.

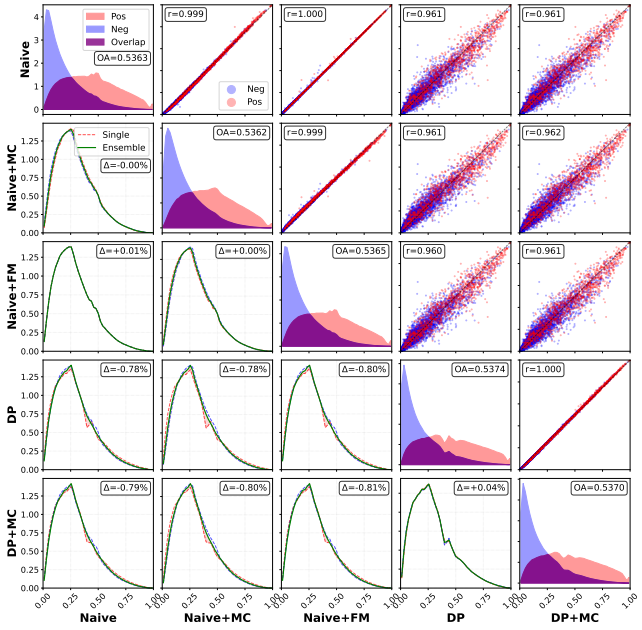


Figure 7: Pairwise prediction comparison matrix (Criteo-Wukong). Diagonal: single-model positive/negative distributions with Overlap Area (OA). Upper triangle: prediction correlation (r). Lower triangle: overlap change after pairwise ensemble (Δ, negative indicates improvement).

Performance Validation. Figure 8 further validates these above-mentioned observations. Training-free combinations show minimal AUC changes compared to baseline, while cross-training combinations achieve significant improvements. Notably, combining identical training strategies (DP+DP MC) degrades performance, confirming that ensemble gains require diverse training sources. These negative experiments indicate that is diversified training is necessary for test-time scaling for large-scale recommendation.

Why Training Matters. Here we provide a hypothesis regarding the failure of training-free methods in large-scale recommendation. This can be explained through error decomposition: $E = A + \epsilon$, where A represents *systematic bias* and ϵ captures *random perturbations*. MC-Dropout and Feature Masking only perturb ϵ through

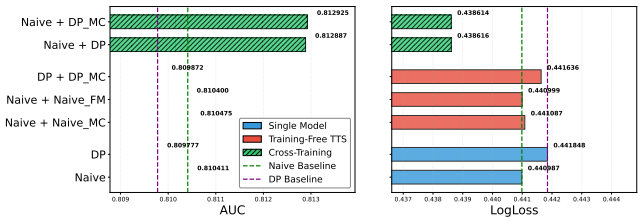


Figure 8: Performance comparison of training-free vs. training-based combinations (Criteo-Wukong). Cross-training combinations (Naive+DP) significantly outperform training-free alternatives (Naive+MC/FM).

stochastic inference but cannot alter the systematic bias A inherited from fixed checkpoints. While averaging among training-free methods reduces random noise ($\mathbb{E}[\epsilon_i] \approx 0$), the systematic error persists, generating correlated predictions with shared underlying bias. This limitation is exacerbated in large-scale recommendation setting, where models develop highly confident predictions with sharply peaked distributions. Stochastic perturbations introduce noise but fail to capture meaningful uncertainty. In contrast, training with different random seeds produces models with diverse systematic biases, whose uncorrelated errors mutually compensate when averaged. While these training-free techniques succeed in high-entropy tasks like language model and sequence recommendation [7, 18], they may not be directly extended to the large-scale recommendation settings, where **effective test-time scaling requires training-based diversity to generate models with uncorrelated systematic biases.**

4 Related Work

4.1 Scaling Laws in Recommendation

Since the exponential growth of LM driven by the scaling law [15], scaling up recommendation models has become a trend in the field [2, 40]. In the recommendation model, these techniques can be categorized into two classes: sequential-based [8, 12, 16, 17, 32, 43, 46, 48] and sequential-free [47, 51]. The sequence-based solutions tend to scale up the length of the sequence [8, 12, 46] or focus on the sequence data [16, 17, 43, 48], while the sequence-free solutions are more focused on the general classification formulation and tend to increase the parameters of the existing model [11, 21, 35, 47, 51]. The sequence-free solutions can be further categorized into depth scaling [47, 51] and width scaling [11, 21, 35]. Note that width scaling shares similarity with existing solutions that utilize various interaction components in recommendation [5, 10, 24, 36, 37]. The major difference lies in the corresponding embedding table, which is used to avoid collisions and achieve better expressiveness [11]. Our paper is more correlated with the sequence-free solutions, as we do not hold any hypothesis on the training corpus. However, our proposed solutions are orthogonal to all existing parameter scaling solutions, as shown empirically in Section 3.4.

4.2 Test-time Scaling

Test-time scaling has consistently become a working solution in trading the inference speed for higher performance [34]. Recently, test-time scaling has become a popular trend in the language model domain [4, 26, 33]. These solutions can generally be decomposed into two dimensions [33, 49]. One is to scale up the length of the output token via solutions such as budget forcing [26]. The other approach is to diversify the output in parallel and generate the final solution via techniques such as self-consistency [39] or best-of-N [4]. Our paper relates to the parallel category, as both of them aim to generate diverse outputs during inference. Specifically, just as best-of-N [4] and self-consistency [39] scale LLM performance by sampling and aggregating multiple outputs, prediction merging scales DLRS performance by training and aggregating multiple models—both increase inference-time compute to improve output quality without modifying the underlying model architecture.

4.3 Ensemble Learning

Prediction merging is conceptually related to ensemble learning, particularly output-level aggregation such as bagging and averaging [20]. Ensemble techniques have been widely adopted in recommendation —ranging from boosting-based cascading [20], knowledge distillation from ensembles [52], to a sparse mixture-of-experts with adaptive gating [24, 44]. Recent width scaling methods [11, 21, 35] can also be viewed as intra-model structural ensembles with shared embedding tables. However, prior work uniformly treats model combination as a static accuracy-boosting technique. Our work shifts the perspective: we systematically study how ensemble performance *scales* with inference-time compute, compare it against parameter scaling under equivalent FLOPs, and analyze its complementarity with model capacity—questions not addressed by the ensemble literature in the recommendation context.

5 Conclusion

In this paper, we systematically investigate the effectiveness of test-time scaling applied in large-scale recommendation systems. We identify the key challenge of applying test-time scaling in recommendation: generating diverse yet meaningful outputs. Hence, we propose two solutions for generating diverse outputs: heterogeneity among different recommendation models and randomness on homogenous architectures. Prediction merging is further proposed as a technique for merging outputs. Through evaluation on three datasets and eight backbones, we demonstrate the scalability of test-time scaling in large-scale recommendation systems. We further demonstrate that test-time scaling consistently achieves higher performance per FLOP and, given sufficient parallel serving resources, lower inference latency than parameter scaling, while also being more stable and orthogonal to parameter scaling. Ablation study further provides intuition on the mechanism behind prediction merging.

Acknowledgments

The authors would like to thank the Shenzhen Technology University for the financial support through the SZTU University Research Project (Grant No. 20251061020002).

References

[1] Aden and Yi Wang. 2012. KDD Cup 2012, Track 2. https://kaggle.com/competitions/kddcup2012-track2. Kaggle.

[2] Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. 2022. Understanding Scaling Laws for Recommendation Models. *CoRR* abs/2208.08489 (2022). arXiv:2208.08489 doi:10.48550/ARXIV.2208.08489

[3] Ryo Bertolissi, Jonas Hübotter, Ido Hakimi, and Andreas Krause. 2025. Local Mixtures of Experts: Essentially Free Test-Time Training via Model Merging. In *The Second Conference on Language Modeling, CoLM 2025*. OpenReview.net, Montreal, Canada. https://openreview.net/forum?id=X2RXpFA6Vh

[4] Bradley C. A. Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. *CoRR* abs/2407.21787 (2024). arXiv:2407.21787 doi:10.48550/ARXIV.2407.21787

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016*. ACM, Boston, MA, USA, 7–10. doi:10.1145/2988450.2988454

[6] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences* (2 ed.). Lawrence Erlbaum Associates, Hillsdale, NJ.

[7] Yizhou Dang, Yuting Liu, Enneng Yang, Minhan Huang, Guibing Guo, Jianzhe Zhao, and Xingwei Wang. 2025. Data Augmentation as Free Lunch: Exploring the Test-Time Augmentation for Sequential Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 1466–1475. doi:10.1145/3726302.3729943

[8] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment. *CoRR* abs/2502.18965 (2025). arXiv:2502.18965 doi:10.48550/ARXIV.2502.18965

[9] Jonathan Frankle and Michael Carbin. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *7th International Conference on Learning Representations, ICLR 2019*. OpenReview.net, New Orleans, LA, USA. https://openreview.net/forum?id=rJl-b3RcF7

[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *26th International Joint Conference on Artificial Intelligence, IJCAI 2017*. ijcai.org, Melbourne, Australia, 1725–1731.

[11] Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. 2024. On the Embedding Collapse when Scaling up Recommendation Models. In *Forty-first International Conference on Machine Learning, ICML 2024*. OpenReview.net, Vienna, Austria. https://openreview.net/forum?id=aPVwOAr1aW

[12] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, Yueming Han, Menglei Zhou, Lei Yu, Chuan Liu, and Wei Lin. 2025. MTGR: Industrial-Scale Generative Recommendation Framework in Meituan. *CoRR* abs/2505.18654 (2025). arXiv:2505.18654 doi:10.48550/ARXIV.2505.18654

[13] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2023. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, Kigali, Rwanda. https://openreview.net/forum?id=6t0Kwf8-jrj

[14] Olivier Chapelle Jean-Baptiste Tien, joycenv. 2014. Display Advertising Challenge. https://kaggle.com/competitions/criteo-display-ad-challenge

[15] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *CoRR* abs/2001.08361 (2020). https://arxiv.org/abs/2001.08361

[16] Kirill Khrylchenko, Artem Matveev, Sergey S. Makeev, and Vladimir Baikalov. 2025. Scaling Recommender Transformers to One Billion Parameters. *CoRR* abs/2507.15994 (2025). arXiv:2507.15994 doi:10.48550/ARXIV.2507.15994

[17] Weijiang Lai, Beihong Jin, Jiongyan Zhang, Yiyuan Zheng, Jian Dong, Jia Cheng, Jun Lei, and Xingxing Wang. 2025. Exploring Scaling Laws of CTR Model for Online Performance Improvement. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems, RecSys 2025*. ACM, Prague, Czech Republic, 114–123. doi:10.1145/3705328.3748046

[18] Andreanne Lemay, Katharina Hoebel, Christopher P. Bridge, Brian Befano, and Silvia De Sanjosé. 2022. Improving the Repeatability of Deep Learning Models with Monte Carlo Dropout. *NPJ Digital Medicine* 5 (Nov. 2022), 174. doi:10.1038/s41746-022-00709-3

[19] 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, KDD 2018*. ACM, London, UK, 1754–1763. doi:10.1145/3219819.3220023

[20] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model Ensemble for Click Prediction in Bing Search Ads. In *Proceedings of the 26th International Conference on World Wide Web Companion* (Perth, Australia) *(WWW '17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. doi:10.1145/3041021.3054192

[21] Xiaolong Liu, Zhichen Zeng, Xiaoyi Liu, Siyang Yuan, Weinan Song, Mengyue Hang, Yiqun Liu, Chaofei Yang, Donghyun Kim, Wen-Yen Chen, Jiyan Yang, Yiping Han, Rong Jin, Bo Long, Hanghang Tong, and Philip S. Yu. 2024. A Collaborative Ensemble Framework for CTR Prediction. *CoRR* abs/2411.13700 (2024). arXiv:2411.13700 doi:10.48550/ARXIV.2411.13700

[22] Fuyuan Lyu, Xing Tang, Dugang Liu, Liang Chen, Xiuqiang He, and Xue Liu. 2023. Optimizing Feature Set for Click-Through Rate Prediction. In *Proceedings of the ACM Web Conference 2023, WWW 2023*. ACM, Austin, TX, USA, 3386–3395. doi:10.1145/3543507.3583545

[23] Fuyuan Lyu, Xing Tang, Dugang Liu, Haolun Wu, Chen Ma, Xiuqiang He, and Xue Liu. 2023. Feature Representation Learning for Click-through Rate Prediction: A Review and New Perspectives. arXiv:2302.02241 [cs.IR] https://arxiv.org/abs/2302.02241

[24] 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, KDD 2018*. ACM, London, UK, 1930–1939. doi:10.1145/3219819.3220007

[25] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, Washington, DC, USA, 4552–4560. doi:10.1609/AAAI.V37I4.25577

[26] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *CoRR* abs/2501.19393 (2025). doi:10.48550/ARXIV.2501.19393

[27] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *CoRR* abs/1906.00091 (2019). http://arxiv.org/abs/1906.00091

[28] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-Based Neural Networks for User Response Prediction. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, Barcelona, Spain, 1149–1154. doi:10.1109/ICDM.2016.0151

[29] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-Based Neural Networks for User Response Prediction over Multi-Field Categorical Data. *ACM Trans. Inf. Syst.* 37, 1 (2019), 5:1–5:35.

[30] Steffen Rendle. 2010. Factorization Machines. In *ICDM 2010, The 10th IEEE International Conference on Data Mining*. IEEE Computer Society, Sydney, Australia, 995–1000. doi:10.1109/ICDM.2010.127

[31] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *5th International Conference on Learning Representations, ICLR 2017*. OpenReview.net, Toulon, France. https://openreview.net/forum?id=B1ckMDqlg

[32] Kyuyong Shin, Hanock Kwak, Su Young Kim, Max Nihlén Ramström, Jisu Jeong, Jung-Woo Ha, and Kyung-Min Kim. 2023. Scaling Law for Recommendation Models: Towards General-Purpose User Representations. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*. AAAI Press, Washington, DC, USA, 4596–4604. doi:10.1609/AAAI.V37I4.25582

[33] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM Test-Time Compute Optimally can be More Effective than caling Model Parameters. *CoRR* abs/2408.03314 (2024). doi:10.48550/ARXIV.2408.03314

[34] Richard Sutton. 2019. The bitter lesson. *Incomplete Ideas (blog)* 13, 1 (2019), 38.

[35] Jiancheng Wang, Mingjia Yin, Junwei Pan, Ximei Wang, Hao Wang, and Enhong Chen. 2025. Enhancing CTR Prediction with De-correlated Expert Networks. *CoRR* abs/2505.17925 (2025). arXiv:2505.17925 doi:10.48550/ARXIV.2505.17925

[36] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *Proceedings of the ADKDD'17*. ACM, Halifax, NS, Canada, 12:1–12:7. doi:10.1145/3124749.3124754

[37] Ruoxi Wang, Rakesh Shivanna, Derek Zhiyuan Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed H. Chi. 2021. DCN V2: Improved Deep & Cross Network and Practical Lessons for Web-scale Learning to Rank Systems. In *WWW '21: The Web Conference 2021*. ACM / IW3C2, Virtual Event / Ljubljana, Slovenia, 1785–1797. doi:10.1145/3442381.3450078

[38] Steve Wang and Will Cukierski. 2014. Click-Through Rate Prediction. https://kaggle.com/competitions/avazu-ctr-prediction. Kaggle.

[39] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net, Kigali, Rwanda. https://openreview.net/forum?id=1PL1NIMMrw

[40] Yunli Wang, Zixuan Yang, Zhen Zhang, Zhiqiang Wang, Jian Yang, Shiyang Wen, Peng Jiang, and Kun Gai. 2024. Scaling Laws for Online Advertisement Retrieval. *CoRR* abs/2411.13322 (2024). arXiv:2411.13322 doi:10.48550/ARXIV.2411.13322

[41] Yejing Wang, Xiangyu Zhao, Tong Xu, and Xian Wu. 2022. AutoField: Automating Feature Selection in Deep Recommender Systems. In *WWW '22: The ACM Web Conference 2022*. ACM, Virtual Event, Lyon, France, 1977–1986. doi:10.1145/3485447.3512071

[42] Prateek Yadav, Derek Tam, Leshem Choshen, Colin A. Raffel, and Mohit Bansal. 2023. TIES-Merging: Resolving Interference When Merging Models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*. Curran Associates, New Orleans, LA, USA. http://papers.nips.cc/paper_files/paper/2023/hash/1644c9af28ab7916874f6fd6228a9bcf-Abstract-Conference.html

[43] Bencheng Yan, Shilei Liu, Zhiyuan Zeng, Zihao Wang, Yizhen Zhang, Yujin Yuan, Langming Liu, Jiaqi Liu, Di Wang, Wenbo Su, Pengjie Wang, Jian Xu, and Bo Zheng. 2025. Unlocking Scaling Law in Industrial Recommendation Systems with a Three-step Paradigm based Large User Model. *CoRR* abs/2502.08309 (2025). arXiv:2502.08309 doi:10.48550/ARXIV.2502.08309

[44] YaChen Yan and Liubo Li. 2023. AdaEnsemble: Learning Adaptively Sparse Structured Ensemble Network for Click-Through Rate Prediction. arXiv:2301.08353 [cs.IR] https://arxiv.org/abs/2301.08353

[45] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. 2024. Model Merging in LLMs, MLLMs, and Beyond: Methods, Theories, Applications and Opportunities. *CoRR* abs/2408.07666 (2024). doi:10.48550/ARXIV.2408.07666

[46] 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. 2024. Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations. In *Forty-first International Conference on Machine Learning, ICML 2024*. OpenReview.net, Vienna, Austria. https://openreview.net/forum?id=xye7iNsgXn

[47] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Shen Li, Yanli Zhao, Yuchen Hao, Yantao Yao, Ellie Dingqiao Wen, Jongsoo Park, Maxim Naumov, and Wenlin Chen. 2024. Wukong: Towards a Scaling Law for Large-Scale Recommendation. In *Forty-first International Conference on Machine Learning, ICML 2024*. OpenReview.net, Vienna, Austria. https://openreview.net/forum?id=8iUgr2nuwo

[48] Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2024. Scaling Law of Large Sequential Recommendation Models. In *Proceedings of the 18th ACM Conference on Recommender Systems, RecSys 2024*. ACM, Bari, Italy, 444–453. doi:10.1145/3640457.3688129

[49] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. 2025. What, How, Where, and How Well? A Survey on Test-Time Scaling in Large Language Models. *CoRR* abs/2503.24235 (2025). arXiv:2503.24235 doi:10.48550/ARXIV.2503.24235

[50] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data - - A Case Study on User Response Prediction. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016*, Vol. 9626. Springer, Padua, Italy, 45–57. doi:10.1007/978-3-319-30671-14

[51] Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, Huizhi Yang, Zheng Chai, Zhe Chen, Yuchao Zheng, Qiwei Chen, Feng Zhang, Xun Zhou, Peng Xu, Xiao Yang, Di Wu, and Zuotao Liu. 2025. RankMixer: Scaling Up Ranking Models in Industrial Recommenders. *CoRR* abs/2507.15551 (2025). arXiv:2507.15551 doi:10.48550/ARXIV.2507.15551

[52] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR Prediction via Knowledge Distillation. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*. ACM, Virtual Event, Ireland, 2941–2958. doi:10.1145/3340531.3412704