Wukong: Towards a Scaling Law for Large-Scale Recommendation

Buyun Zhang [*1] **Liang Luo** [*1] **Yuxin Chen** [*1]
Jade Nie [1] **Xi Liu** [1] **Shen Li** [1] **Yanli Zhao** [1]
Yuchen Hao [1] **Yantao Yao** [1] **Ellie Dingqiao Wen** [1] **Jongsoo Park** [1]
Maxim Naumov [1] **Wenlin Chen** [1]

Abstract

Scaling laws play an instrumental role in the sustainable improvement in model quality. Unfortunately, recommendation models to date do not exhibit such laws similar to those observed in the domain of large language models, due to the inefficiencies of their upscaling mechanisms. This limitation poses significant challenges in adapting these models to increasingly more complex real-world datasets. In this paper, we propose an effective network architecture based purely on stacked factorization machines, and a synergistic upscaling strategy, collectively dubbed Wukong, to establish a scaling law in the domain of recommendation. Wukong's unique design makes it possible to capture diverse, any-order of interactions simply through taller and wider layers. We conducted extensive evaluations on six public datasets, and our results demonstrate that Wukong consistently outperforms state-of-the-art models quality-wise. Further, we assessed Wukong's scalability on an internal, large-scale dataset. The results show that Wukong retains its superiority in quality over state-of-the-art models, while holding the scaling law across two orders of magnitude in model complexity, extending beyond 100 GFLOP/example, where prior arts fall short.

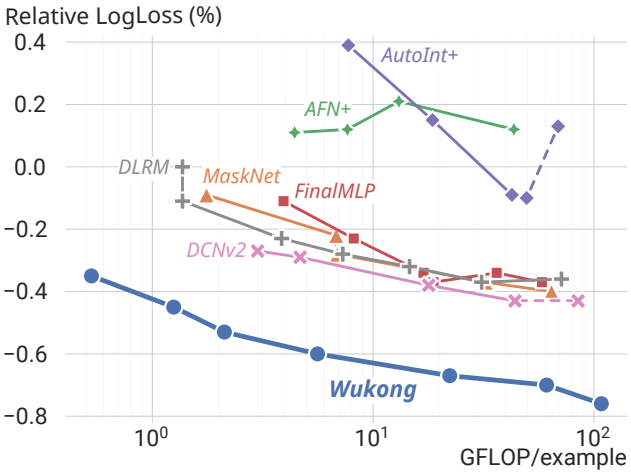

Figure 1: Wukong outperforms existing state-of-the-art models while demonstrating a scaling law in the recommendation domain across two orders of magnitude in model complexity, extending beyond 100 GFLOP/example.

1. Introduction

Deep learning-based recommendation systems (DLRS) power a wide range of online services today (Naumov et al., 2019; Wang et al., 2021a; Lian et al., 2021; Liu et al., 2022; Covington et al., 2016).

Modern DLRS are designed to process a blend of continuous dense features, such as date, and categorical sparse features, like user clicked posts history. Each sparse feature is transformed into a dense embedding representation through a trainable embedding lookup table. These dense embeddings are then fed into an interaction component, designed to capture the intricate interactions between features.

While existing models demonstrate promising accuracy on smaller datasets, their capability to adapt to the scale and intricacy of substantially larger datasets, and to sustain continuous quality improvement as these models scale up, remains less certain. This scalability is increasingly crucial, as modern datasets have seen exponential growth. For example, production datasets today might contain hundreds of billions of training examples (Wang et al., 2021a). Furthermore, foundational models (Bommasani et al., 2021) need to operate at scale to handle larger and multiple complex input sources at the same time. Thus, the need for a DLRS that can both upscale and downscale effectively, adjusting to varying dataset sizes and computational constraints, is paramount. This scalability is encompassed in

[*]Equal contribution [1]Meta AI. Correspondence to: Buyun Zhang <buyunz@meta.com>, Liang Luo <liangluo@meta.com>, Yuxin Chen <yuxinc@meta.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

what is known as a "scaling law" (Kaplan et al., 2020).

To date, the primary trend of DLRS up-scaling is through *sparse scaling*, i.e., expanding the sizes of embedding tables (more rows and/or higher dimensions) for less collision and better expressiveness. Consequently, DLRS have reached trillions of parameters (Kang et al., 2020; Mudigere et al., 2021; Lian et al., 2021) with embedding tables dominating the parameter count. Unfortunately, this traditional way of up-scaling has a few practical drawbacks. Merely expanding the sparse component of a model does not enhance its ability to capture the complex interactions among an increasing number of features. Moreover, this trend notably diverges from the trend of hardware advancements, as most improvements in the next generation accelerators lie in the compute capacity (Luo et al., 2018; 2017), which embedding table lookups cannot utilize. Thus, simply expanding embedding table leads to prohibitive infrastructure costs with suboptimal accelerator utilization, especially in distributed settings (Luo et al., 2024).

Our work aims to find an alternative scaling mechanism for recommendation models, that can establish a scaling law, similar to that established in the LLM domain. Namely, we would like to devise a unified architecture whose quality can be continuously improved in conjunction with dataset size, compute and parameter budgets, with a synergistic strategy.

We focus on upscaling interaction components, dubbed *dense scaling*, to mitigate the quality and efficiency drawbacks from sparse scaling. However, existing models cannot benefit from this paradigm for various reasons. For example, DLRM lacks the ability to capture <higher-order> interactions; DCNv2 and AutoInt+ lack strategy for effective upscaling, leading to rapidly diminishing returns when scaling up; further, even with modern tricks like residual connection (He et al., 2016), layernorm (Ba et al., 2016), gradient clip (Pascanu et al., 2013)), up-scaling existing models is prone to training stability issues (Tang et al., 2023).

To establish a scaling law for recommendation models, we propose Wukong, a simple interaction architecture that exhibits effective dense scaling properties. Inspired by the principles of binary exponentiation, our key innovation is to use a series of stacked Factorization Machines (FMs) to efficiently and scalably capture any-order feature interactions. In our design, each FM is responsible of capturing second order interactions with respect to its inputs, and the outputs from these FMs are subsequently transformed by MLPs into new embeddings, which encode the interactions results and serve as inputs to the next layers.

We evaluated Wukong's performance using six public datasets and a large-scale internal dataset. The results demonstrate that Wukong outperforms state-of-the-art models across all public datasets in terms of AUC, indicating the effectiveness of Wukong's architecture and its ability to generalize across a wide range of recommendation tasks and datasets. In our internal dataset, Wukong not only significantly outperforms existing models in terms of quality at comparable levels of complexity but also shows continuous enhancements in quality when scaled up across two orders of magnitude in model complexity, extending beyond 100 GFLOP/example, where prior arts fall short.

2. Related Work

Deep Learning Recommendation Systems (DLRS) Existing DLRS share a similar structure. A typical model consists of a sparse and a dense component. The sparse component is essentially embedding lookup tables that transform sparse categorical features into dense embeddings, whereas the dense component is responsible for capturing interactions among these embeddings to generate a prediction.

Dense Interaction Architectures Capturing interaction between features is the key to DLRS effectiveness, and we highlight some of the prior arts. AFN+ (Cheng et al., 2020) transforms features into a logarithmic space to capture arbitrary order of interactions; AutoInt+ (Song et al., 2019) uses multi-head self-attention; DLRM and DeepFM (Naumov et al., 2019; Guo et al., 2017) leverage Factorization Machines (FM) (Rendle, 2010) to explicitly capture second order interactions; HOFM (Blondel et al., 2016) optimizes FM to efficiently capture higher order of interactions; DCNv2 (Wang et al., 2021a) uses CrossNet, which captures interactions via stacked feature crossing, which can be viewed as a form of elementwise input attention. FinalMLP (Mao et al., 2023) employs a bilinear fusion to aggregate results from two MLP streams, each takes stream-specific gated features as input. MaskNet (Wang et al., 2021b) adopts a series of MaskBlocks for interaction capture, applying "input attention" to the input itself and intermediate activations of DNN; xDeepFM (Lian et al., 2018) combines a DNN with a Compressed Interaction Network, which captures interactions through outer products and compressing the results with element-wise summation.

Scaling up DLRS (Kang et al., 2020; Mudigere et al., 2021; Lian et al., 2021) provides mechanisms on sparse scaling. (Shin et al., 2023) focuses on scaling up user representation models, with the largest model reported having a total compute less than 0.1 PF-days, (Zhang et al., 2023) aims to improve sequence modeling on the user side, with the largest model reported having less than 0.8B parameters. Additionally (Ardalani et al., 2022) studied the scaling law of DLRM, which is incorporated as a baseline in our work and further scaled up in our experiments. Orthogonally, (Zhao et al., 2023b) proposes a user-centric ranking formulation to improve scalability; (Guo et al., 2023) provided insights on sparse scaling, demonstrating limits on prior

arts and is complement to our work. Further, VIP5 (Geng et al., 2023) leverages existing scaling laws in LLMs to apply a multimodal LLM to recommendation, however, (Lin et al., 2023) points out that further study is needed to verify whether larger implies better in LLM-powered recommenders, while (Huang et al., 2024) suggests evaluations on more diverse datasets are needed to for a conclusion.

3. Design of Wukong

We keep two objectives in mind when designing Wukong's architecture: (1) to effectively capture the intricate high-order feature interactions; and (2) to ensure Wukong's quality scale gracefully with respect to dataset size, GFLOP/example and parameter budgets.

3.1. Overview

In Wukong, categorical and dense features initially pass through an **Embedding Layer** (Sec. 3.2), which transforms these inputs into *Dense Embeddings*.

As shown in Figure 2, Wukong subsequently adopts an **Interaction Stack** (Sec. 3.3), a stack of unified neural network layers to capture the interaction between embeddings. The Interaction Stack draws inspiration from the concept of binary exponentiation, allowing each successive layer to capture exponentially higher-order interactions. Each layer in the Interaction Stack consists of a **Factorization Machine Block** (FMB, Sec. 3.4) and a **Linear Compression Block** (LCB, Sec. 3.5). FMB and LCB independently take in input from last layer and their outputs are ensembled as the output for the current layer. Following the interaction stack is a final Multilayer Perceptron (MLP) layer that maps the interaction results into a prediction.

3.2. Embedding Layer

Given a multi-hot categorical input, an embedding table maps it to a dense embedding. This process involves a series of lookups, each corresponding to a "hot" dimensions within the input. The lookup results are then aggregated using a pooling operation (usually a summation).

In our design, the embedding dimension is standardized for all embeddings generated by the Embedding Layer, known as the global embedding dimension d. To accommodate the varying significance of different features, multiple embeddings are generated for each feature that is deemed significant. Less important features are allocated smaller underlying embedding dimensions. These smaller embeddings are then collectively grouped, concatenated, and transformed into d-dimensional embeddings using a MLP.

Dense inputs are transformed by an MLP into latent embeddings that share the same d dimension, and are joined with the embedding outputs of categorical input. This yields an output tensor of size $X_0 \in \mathbb{R}^{n \times d}$, where n is the total number of embeddings from the dense and sparse part. X_0 is then ready to be further processed by the Interaction Stack.

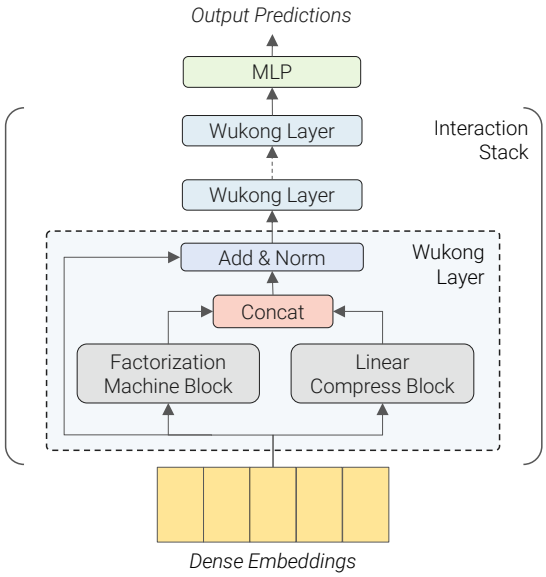

Figure 2: Wukong employs an interaction stack to capture feature interactions. Each layer in the stack consists of a Factorization Machine Block and a Linear Compress Block.

Note that unlike conventional approaches like DCN (Wang et al., 2021a), we interpret each embedding vector as a whole unit (detailed later), and hence our representation of $X_0 \in \mathbb{R}^{n \times d}$ as opposed to $X_0 \in \mathbb{R}^{nd}$.

3.3. Interaction Stack

The interaction modules stack l identical interaction layers, where each layer captures progressively higher-order feature interactions using Factorization Machines (FMs).

An interaction layer has two blocks in parallel: a Factorization Machine Block (FMB) and a Linear Compression Block (LCB). FMB computes feature interactions between input embeddings of the layer, and LCB simply forwards linearly compressed input embeddings of the layer. The outputs of FMB and LCB are then concatenated.

For layer i in the stack, its results can contain feature interactions with arbitrary order from 1 to 2^i. This can be simply shown by induction. Let's assume the input of layer i contains interactions of order from 1 to 2^{i-1}, which is true for the first layer (i.e. $i = 1$). Since FMB generates $(o_1 + o_2)$-order feature interactions given $o1$ and $o2$-order interactions, then we have immediately the output of layer i containing 1 to 2^i-order interactions, with the lower bound achieved from the output of LCB and the upper bound achieved by the FM

interacting two 2^{i-1}-order interactions from the input.

To help stabilize training, we also adopt residual connections across layers, followed by layer normalization (LN). Putting everything together, we have

$$X_{i+1} = \text{LN}(\text{concat}(\text{FMB}_\text{i}(X_i), \text{LCB}_\text{i}(X_i)) + X_i)$$

Depending on the specific configurations of FMB and LCB, X_i may have a different number of embeddings than X_{i+1}, which usually happens at the first layer. To handle this case, the residual can be linearly compressed to match the shape.

3.4. Factorization Machine Block (FMB)

A FMB contains a FM followed by a MLP. The FM is used to capture explicit feature interactions of the input embeddings, with the output being a 2D interaction matrix where each element represents the interaction between a pair of embeddings. This interaction matrix is flattened and converted to a vector with shape of $(n_F \times d)$ through the MLP, and reshaped to n_F embeddings for later use.

Operationally, a FMB does the following:

$$\text{FMB}(X_i) = \text{reshape}(\text{MLP}(\text{LN}(\text{flatten}(\text{FM}(X_i)))))$$

Wukong's FM module is fully customizable: for example, in the most basic version, we followed the FM design in (Naumov et al., 2019), i.e., taking the dot product between all pairs of embedding vectors, $FM(X) = XX^T$. We discuss more optimized FM designs in Sec. 3.6.

3.5. Linear Compress Block (LCB)

LCB simply linearly recombines embeddings without increasing interaction orders, which is critical in ensuring that the invariance of interaction order is maintained throughout the layers. Specifically, it guarantees that the i-th interaction layer captures interaction orders ranging from 1 to 2^i. The operation performed by a LCB can be described as follows:

$$\text{LCB}(X_i) = W_L X_i$$

where $W_L \in \mathbb{R}^{n_L \times n_i}$ is a weight matrix, n_L is a hyperparameter indicating the number of compressed embeddings, and n_i is the number of input embeddings of layer i.

3.6. Optimized FM

FM's computation and storage complexity grows quadratically with the number of embeddings with the pair-wise dot product, and This quickly becomes prohibitive on real-world datasets with thousands of features.

To allow effective feature interaction while lowering compute cost, we adopt a similar scheme to (Sharma, 2023; Anonymous, 2019) that leverage low-rank property in pair-wise dot product matrix, which was observed in many real-world datasets (Wang et al., 2021a).

When $d <= n$, the dot-product interaction XX^T is a d-rank matrix, which is often the case on large datasets whose number of features is larger than the embedding dimension. Therefore, we can effectively reduce the size of output matrix from $n \times n$ to $n \times k$, where k is a hyperparameter, by multiplying XX^T with a learnable projection matrix Y of shape $n \times k$ (i.e., computing XX^TY) without loss of information in theory. This reduces memory requirement to store the interaction matrix. We can then take advantage of the associative law to compute X^TY first, further reducing compute complexity from $O(n^2d)$ to $O(nkd)$ with $k << n$.

Furthermore, to enhance the model quality, the projection matrix Y can be made attentive to the input by processing linearly compressed input through a MLP. We use the optimized FM in our following experiments by default, unless mentioned otherwise.

3.7. Complexity Analysis

We assume each layer in the Interaction Stack uses the same hyperparameters, and the largest FC in the MLP has size h.

For the first layer, the time complexity of FMB is the sum of the FM and the MLP, which is $O(nkd) \approx O(ndh)$ and $O(nkh + h^2 + n_F dh) \approx O(ndh + h^2)$, respectively. The time complexity of LCB is $O(nn_L d) \approx O(ndh)$. For subsequent layers, the time complexity is $O(n'dh + h^2)$, where $n' = n_L + n_F$. Hence, the total time complexity of Wukong is $O(ndh + ln'dh + h^2) \approx O(ndhlogn + h^2)$.

3.8. Scaling Wukong

We now summarize the main hyperparameters that are related to scale up and later we describe our efforts to upscaling Wukong with respect to these hyperparameters.

- l: number of layers in the Interaction Stack.
- n_F: number of embeddings generated by FMB
- n_L: number of embeddings generated by LCB
- k: number of compressed embeddings in optimized FM
- MLP: number of layers and FC size in the MLP of FMB

During scaling up, we initially focus on increasing l to enable the model to capture higher-order interactions. Following this, we enlarge other hyperparameters to augment the model's capacity of capturing broader range of interactions.

3.9. Intuition Behind Wukong's Enhanced Effectiveness

Compared to existing work using FM as their primary interaction architecture, Wukong's innovative approach of

stacking FMs greatly enhances the conventional FM's capability. This allows Wukong to capture interactions of any order, making it highly effective for large-scale, complex datasets that require higher-order reasoning. While there are efforts towards high-order FM, Wukong's exponential rate of capturing high-order interactions offers great efficiency, bypassing the linear complexity seen in HOFM and avoiding the costly outer product in xDeepInt.

While MLPs have shown limitations in implicitly capturing interactions (Beutel et al., 2018), Wukong diverges from approaches that rely on MLPs for interaction capture. Instead, Wukong primarily employs MLPs to transform the results of interactions into embedding representations, which are then used for further interactions. This distinct use of MLPs enhances the model's ability to process and interpret complex, heterogeneous features effectively.

Additionally, Wukong treats each embedding as a single unit, focusing on embedding-wise interactions. This approach significantly reduces computational demands compared to architectures that capture element-wise interactions.

4. Implementation

This section discusses practices to effectively train high-complexity Wukong on large-scale datasets.

Overall, distributed training is required to make Wukong training feasible. For the embedding layer, we use a column-wise sharded embedding bag implementation provided by Neo (Mudigere et al., 2021) and NeuroShard (Zha et al., 2023). On the dense part, we balance the trade-off between performance and memory capacity by adopting FSDP (Zhao et al., 2023a) and tune the sharding factor so that the model fits in the memory without creating too much redundancy.

To enhance training efficiency, we employ automatic operator fusion through to improve training performance. In addition, we aggressively apply quantization to reduce compute, memory, and communication overheads simultaneously. Specifically, we train Wukong's embedding tables in FP16, and communicate embedding lookup results in FP16 in the forward pass and BF16 in the backward pass; we use BF16 quantization during the transport gradients for dense parameters in the backward pass.

5. Overview of Evaluations

We evaluate Wukong using six public datasets and an internal dataset, details of which are summarized in Table 1. The results of these evaluations are organized in two sections.

In Section 6, we evaluate on six public datasets, focusing on demonstrating the effectiveness of Wukong in the low complexity realm. Our results show that **Wukong surpasses previous state-of-the-art methods across all six datasets, demonstrating its effectiveness.**

	#Samples	#Features
Frappe	0.29M	10
MicroVideo	1.7M	7
MovieLens Latest	2M	3
KuaiVideo	13M	8
TaobaoAds	26M	21
Criteo Terabyte	4B	39
Internal	146B	720

Table 1: Statistics of our evaluation datasets.

In Section 7, we evaluate on our large-scale in-house dataset to demonstrate the scalability of Wukong. The dataset contains 30 times more samples and 20 times more features compared to one of the largest dataset Criteo. Our results reveals that **(1) Wukong consistently outperforms all baseline models in terms of both model quality and runtime speed, maintaining this superiority across all complexity scales; (2) Wukong exhibits a better scaling trend in comparison to baseline models.** We also conduct an ablation study to gain understanding of the individual contributions and the effectiveness of each component within Wukong.

6. Evaluation on Public Datasets

In this section, we aim to demonstrate the effectiveness of Wukong across a variety of public datasets. Unless noted otherwise, we use the preproc provided by the BARS benchmark (Zhu et al., 2022b) for consistency with prior work.

6.1. General Evaluation Setup

6.1.1. Datasets

Frappe (Baltrunas) is an app usage log. This datasets predicts whether a user uses the app with the given contexts.

MicroVideo (Chen et al., 2018) is a content understanding-based dataset provided by THACIL work containing interactions between users and micro-videos. This log contains multimodal embeddings, together with traditional features.

MovieLens Latest (Harper & Konstan, 2015) is a well known dataset that contains users' ratings on movies.

KuaiVideo (Kuaishou) is the competition dataset released by Kuaishou. The dataset is used to predict the click probability of a user on new micro-videos. This dataset also contains content understanding-based embeddings along with other categorical and float features.

TaobaoAds (Tianchi, 2018) This dataset includes 8 days of ads click through rate (CTR) prediction on Taobao.

	Frappe		MicroVideo		MovieLens L.		KuaiVideo		TaobaoAds		Criteo TB	
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
Baselines												
AFN+	0.9812	0.2340	0.7220	0.4142	0.9648	0.3109	0.7348	0.4372	0.6416	0.1929	0.8023	0.1242
AutoInt+	0.9806	0.1754	0.7155	0.4203	0.9693	0.2178	0.7297	0.4376	0.6437	0.1930	0.8073	0.1233
DCNv2	0.9774	0.2325	0.7187	0.4162	0.9683	0.2169	0.7360	0.4383	0.6457	0.1926	0.8096	0.1227
DLRM	0.9846	0.1465	0.7173	0.4179	0.9685	0.2160	0.7357	0.4382	0.6430	0.1931	0.8076	0.1232
FinalMLP	**0.9868**	0.1280	0.7247	0.4147	**0.9723**	0.2211	0.7374	0.4435	0.6434	0.1928	0.8096	0.1226
MaskNet	0.9816	0.1701	0.7255	0.4157	0.9676	0.2383	0.7376	0.4372	0.6433	0.1927	0.8100	0.1227
xDeepFM	0.9780	0.2441	0.7167	0.4172	0.9667	0.2089	0.7118	0.4565	0.6342	0.1961	0.8084	0.1229
Ours												
Wukong	**0.9868**	0.1757	**0.7292**	0.4148	**0.9723**	0.1794	**0.7414**	0.4367	**0.6488**	0.1954	**0.8106**	0.1225

Table 2: Evaluation results on six public datasets. The model with **best AUC** and best LogLoss on each dataset are highlighted.

Criteo Terabyte (Criteo) This dataset contains 24 days of ads click feedback. We used the last day of data for testing.

6.1.2. Baselines

We benchmark Wukong against seven widely recognized state-of-the-art models used in both academia and industry, including AFN+ (Cheng et al., 2020), AutoInt+ (Song et al., 2019), DLRM (Naumov et al., 2019), DCNv2 (Wang et al., 2021a), FinalMLP (Mao et al., 2023), MaskNet (Wang et al., 2021b) and xDeepFM (Lian et al., 2018).

6.1.3. Metrics

AUC Area Under the Curve (AUC) measures the model's ability to correctly classify positives and negatives across all thresholds. Higher the better. We use AUC as the basis for hyperparameter tuning and topline metric for reporting, following recommendation conventions (Tien et al., 2014; Blondel et al., 2016; Song et al., 2019; Wang et al., 2021a; Zhu et al., 2022b; Mao et al., 2023).

LogLoss The log loss quantifies the penalty based on how far the prediction is from the actual label. Lower the better.

6.2. Model-Specific Setup

For the five smaller datasets, aside from Criteo, we adopted the public BARS evaluation framework (Zhu et al., 2022a; 2021). We directly use the best searched model configs on BARS whenever possible, and use the provided model default hyperparameters for the rest. In addition to the default embedding dimension provided in the framework, we further test an embedding dimension of 128 and report whichever of these two configurations yielded better results. For Wukong, we tune the dropout rate and optimizer settings and compression of LCB to adapt to the number of features.

We leverage the larger Criteo dataset to evaluate the model performance on realistic online recommendation systems, where one-pass training is performed. In light of the new training setup, we conducted extensive grid search using the system described in Sec. 4 for all baselines and Wukong to facilitate fair comparisons. This exhaustive process involved nearly 3000 individual runs. We provide the model-specific search space in Appendix A. The best searched model hyperparameters were later used as the base config in Sec 7.

6.3. Results

We summarize the results in Table 2. Overall, Wukong is able to achieve state-of-the-art results in terms of AUC across all public datasets. This result demonstrates the effectiveness of Wukong's architecture and its ability to comprehend diverse datasets and to generalize across a wide range of recommendation tasks.

7. Evaluation on an Internal Dataset

In this section, we show the scalability of Wukong and gain a deep understanding of how different individual components of Wukong contribute to its effectiveness, using a large-scale dataset which enables the study for merging properties that is not seen in small, public datasets.

7.1. Evaluation Setup

7.1.1. Dataset

This dataset contains 146B entries in total and has 720 distinct features. Each feature describes a property of either the item or the user. There are two tasks associated with this dataset: *(Task1)* predicting whether a user has showed interested in an item (e.g., clicked) and *(Task2)* whether a conversion happened (e.g., liked, followed).

7.1.2. Metrics

GFLOP/example Giga Floating Point Operations per example (GFLOP/example) quantifies the computational complexity during model training.

PF-days The total amount of training compute equivalent to running a machine operating at 1 PetaFLOP/s for 1 day.

#Params Model size measured by the number of parameters in the model. The sparse embedding table size was fixed to 627B parameters.

Relative LogLoss LogLoss improvement relative to a fixed baseline. We opt to use the DLRM with the basic config as the baseline. A 0.02% Relative LogLoss improvement is considered as significant on this dataset. We report relative LogLoss on the last 1B-window during online training.

7.1.3. Baselines

We adhere to the same baseline setup as detailed in Sec. 6.1.2. However, xDeepFM was not included in the reported results, due to the incompatibility of its expensive outer product operation with the large-scale dataset, consistently causing out-of-memory issues even in minimal setups.

7.1.4. Training

We used the best optimizer configuration found in our pilot study across all experiments, i.e., Adam with lr=0.04 with beta1=0.9, beta2=1 for dense part and Rowwise Adagrad with lr=0.04 for sparse embedding tables. Models were trained and evaluated in an online training manner. We fix the embedding dimension to 160 across all runs.

We set the hyperparameters with the best configuration found on the Criteo Terabyte evaluation described in Sec. 6 as a starting point, and gradually scale up parameter count for each model. We use a global batch size of 262,144 for all experiments. Each experiment was run on 128 or 256 H100 GPUs depending on the model size.

7.2. Results

We observed comparable results for both tasks, and report results for *Task1* in the main text, while the detailed results of *Task2* are provided in Appendix C.

Quality vs. Compute Complexity In Fig. 1, we depict the relationship between quality and compute complexity (empirically, $y = -100 + 99.56x^{0.00071}$). The results show that Wukong consistently outperforms all baselines across various complexity levels, achieving over 0.2% improvement in LogLoss. Notably, Wukong holds its scaling law across two orders of magnitude in model complexity – approximately translating to a 0.1% improvement for every quadrupling of complexity. Among baselines, AFN+, DLRM and FinalMLP tend to reach a plateau after a certain complexity level, while AutoInt+, DCNv2 and MaskNet failed to further enhance quality [1]. Nonetheless, even DCNv2, the top-performing baseline, demands a 40-fold increase in complexity to match Wukong's quality.

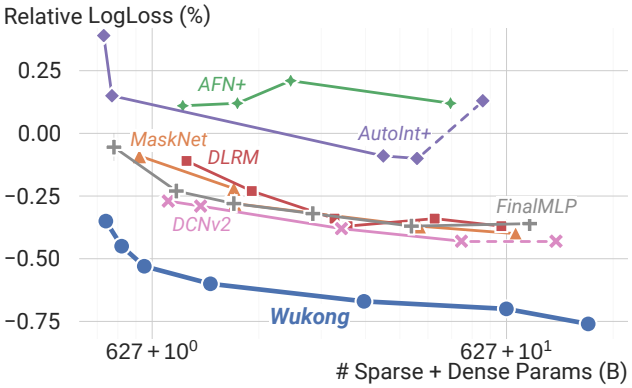

Figure 3: Scalability of Wukong with respect to # parameters on the internal dataset.

Quality vs. Model Size In Fig. 3, we illustrate the correlation between model quality and model size. Echoing the trends observed in compute complexity scaling above, Wukong consistently outperforms all baselines by roughly 0.2% across all scales of model size. while demonstrating a steady improvement trend up to over 637 billion parameters[2]

Quality vs. Data Size See Appendix E.

Model-Specifc Scaling Throughout the scaling process, we employed distinct strategies per model. Detailed hyperparameter settings for each run are provided in Appendix C. Scaling processes of each model are summarized as follows:

Wukong We scaled up Wukong by tuning the hyperparameters detailed in Sec. 3.8.

AFN+ We scaled up AFN's hidden layers, ensemble DNN, and the number of logarithmic neurons. The results show that scaling up AFN does not improve model quality.

AutoInt+ We scaled up multi-head attention and the ensemble DNN. Model quality of this model is initially worse than others, but improves notably when scaling up.

DLRM We scaled up the top MLP. The results show that the quality starts saturated beyond 31 GFLOP/example.

DCNv2 We scaled up both Cross Network and Deep Network. Scaling up Cross Network did not yield any quality improvement. The training stability of DCNv2 is worse than other models and we applied strict gradient clipping.

[1] AutoInt+ and DCNv2 consistently faced significant training instability issue when further scaled up. AutoInt+ recovered from loss explosion, albeit with reduced model quality; while DCNv2 failed to recover, and its quality was estimated from performance before the explosion. MaskNet was hindered by excessive memory consumption, leading to out-of-memory errors, blocking further scaling up.

[2] We verified Wukong's effectiveness in both online and offline settings, and for brevity, we focus on reporting offline metrics.

FinalMLP We scaled up the two MLP streams and the Feature Selection modules. The results show that the model quality improves in the low complexity region, but starts to saturate beyond 36 GFLOP/example.

MaskNet We tested both Parallel and Serial MaskNet, and found that the Parallel variant is better. We decreased the initial reduction ratio to ensure the model has a runnable size, and progressively scaled up number of MaskBlocks, the DNN and the reduction ratio.

7.3. Ablation

Significance of Individual Components Our goal is to demonstrate the importance of FMB, LCB and the residual connection in Wukong's Interaction Stack. To this end, we performed experiments in which each component was individually deactivated by zeroing out its results.

As shown in Fig. 4, nullifying FMB results in a large quality degradation. Interestingly, the deactivation of either LCB or the residual leads to only a modest decline in quality, while disabling both causes a substantial degradation. This observation implies that by zero-padding FMB outputs and incorporating a residual connection, LCB can be simplified.

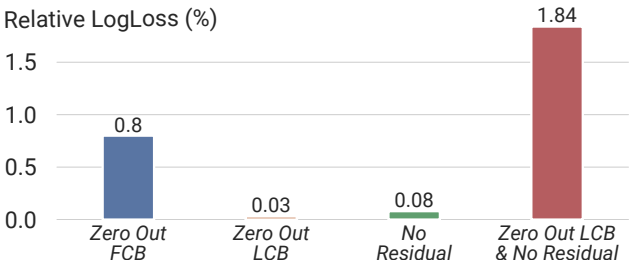

Figure 4: Significance of individual components.

Impact of Scaling Individual Components We aim to dissect the contributions in model quality when scaling up each hyperparameter within Wukong. We started from a base configuration and proceeded to incrementally double each hyperparameter. The results are depicted in Fig. 5. We observed that increasing the number of Wukong layers l leads to a substantial uplift in model quality, due to higher-order interactions being captured. Additionally, augmenting the MLP size results in considerable performance enhancements. Elevating k and n_F proves beneficial, while n_L has plateaued for the base configuration. Notably, a combined scale-up of k, n_F, n_L delivers more pronounced quality improvements than scaling each individually.

8. Discussions

Practically Serving Scaled-up Models Scaling up to high complexity presents notable challenges for real-time serving. Potential solutions include training a multi-task

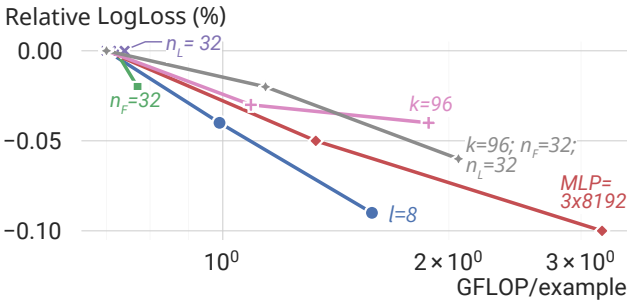

Figure 5: Impact of scaling individual components.

foundation model to amortize costs: distilling knowledge from the large models into small, efficient ones for serving.

Limitation and Future Work We also note limitations and caveats to our work, which can be goals in future work.

Understanding the exact limit of Wukong's scalability is an important area of research. Due to the massive compute requirement, we have not been able to reach a level of complexity where the limit applies.

While Wukong demonstrates superior quality in various evaluations, a comprehensive theoretical understanding of its underlying principles, particularly in contrast to architectures like transformers which share stacked dot product structure, remains an area that needs further exploration.

Additionally, Wukong's generalizability beyond recommendation, particularly in domains that involve heterogeneous input data sources similar to distinct features in recommendation, remains to be further explored and understood.

9. Conclusion

We proposed an effective network architecture, named Wukong. We demonstrated that Wukong establishes a scaling law in the domain of recommendation that is not previously observed – Wukong is able to efficiently scale up and down across two order of magnitude in compute complexity while maintaining a competitive edge over other state of the art models, making it a scalable architecture that can serve as a backbone from small vertical models to large foundational models across a wide range of tasks and datasets.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anonymous. Dot product matrix compression for machine learning. *Technical Disclosure Commons*, 2019.

Ardalani, N., Wu, C.-J., Chen, Z., Bhushanam, B., and Aziz, A. Understanding scaling laws for recommendation models. *arXiv preprint arXiv:2208.08489*, 2022.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Baltrunas, L. Frappe - mobile app usage. URL https://www.baltrunas.info/context-aware.

Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., and Chi, E. H. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 46–54, 2018.

Blondel, M., Fujino, A., Ueda, N., and Ishihata, M. Higher-order factorization machines. *Advances in Neural Information Processing Systems*, 29, 2016.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Chen, X., Liu, D., Zha, Z.-J., Zhou, W., Xiong, Z., and Li, Y. Temporal hierarchical attention at category- and item-level for micro-video click-through prediction. In *MM*, 2018.

Cheng, W., Shen, Y., and Huang, L. Adaptive factorization network: Learning adaptive-order feature interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3609–3616, 2020.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

Criteo. Criteo 1tb click logs dataset. https://ailab.criteo.com/download-criteo-1tb-click-logs-dataset/.

Geng, S., Tan, J., Liu, S., Fu, Z., and Zhang, Y. Vip5: Towards multimodal foundation models for recommendation. *arXiv preprint arXiv:2305.14302*, 2023.

Gui, H., Wang, R., Yin, K., Jin, L., Kula, M., Xu, T., Hong, L., and Chi, E. H. Hiformer: Heterogeneous feature interactions learning with transformers for recommender systems. *arXiv preprint arXiv:2311.05884*, 2023.

Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

Guo, X., Pan, J., Wang, X., Chen, B., Jiang, J., and Long, M. On the embedding collapse when scaling up recommendation models. *arXiv preprint arXiv:2310.04400*, 2023.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5 (4), dec 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Huang, C., Yu, T., Xie, K., Zhang, S., Yao, L., and McAuley, J. Foundation models for recommender systems: A survey and new perspectives. *arXiv preprint arXiv:2402.11143*, 2024.

Kang, W.-C., Cheng, D. Z., Yao, T., Yi, X., Chen, T., Hong, L., and Chi, E. H. Learning to embed categorical features without embedding tables for recommendation. *arXiv preprint arXiv:2010.10784*, 2020.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kuaishou. URL https://www.kuaishou.com/activity/uimc.

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., and Sun, G. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1754–1763, 2018.

Lian, X., Yuan, B., Zhu, X., Wang, Y., He, Y., Wu, H., Sun, L., Lyu, H., Liu, C., Dong, X., Liao, Y., Luo, M., Zhang, C., Xie, J., Li, H., Chen, L., Huang, R., Lin, J., Shu, C., Qiu, X., Liu, Z., Kong, D., Yuan, L., Yu, H., Yang, S., Zhang, C., and Liu, J. Persia: An open, hybrid system scaling deep learning-based recommenders up to 100 trillion parameters. November 2021.

Lin, J., Dai, X., Xi, Y., Liu, W., Chen, B., Li, X., Zhu, C., Guo, H., Yu, Y., Tang, R., et al. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817*, 2023.

Liu, Z., Zou, L., Zou, X., Wang, C., Zhang, B., Tang, D., Zhu, B., Zhu, Y., Wu, P., Wang, K., et al. Monolith: Real

time recommendation system with collisionless embedding table. corr abs/2209.07663 (2022), 2022.

Luo, L., Liu, M., Nelson, J., Ceze, L., Phanishayee, A., and Krishnamurthy, A. Motivating in-network aggregation for distributed deep neural network training. In *Workshop on Approximate Computing Across the Stack*, 2017.

Luo, L., Nelson, J., Ceze, L., Phanishayee, A., and Krishnamurthy, A. Parameter hub: a rack-scale parameter server for distributed deep neural network training. In *Proceedings of the ACM Symposium on Cloud Computing*, pp. 41–54, 2018.

Luo, L., Zhang, B., Tsang, M., Ma, Y., Chu, C.-H., Chen, Y., Li, S., Hao, Y., Zhao, Y., Lakshminarayanan, G., et al. Disaggregated multi-tower: Topology-aware modeling technique for efficient large-scale recommendation. *arXiv preprint arXiv:2403.00877*, 2024.

Mao, K., Zhu, J., Su, L., Cai, G., Li, Y., and Dong, Z. Finalmlp: An enhanced two-stream mlp model for ctr prediction. *arXiv preprint arXiv:2304.00902*, 2023.

Mudigere, D., Hao, Y., Huang, J., Tulloch, A., Sridharan, S., Liu, X., Ozdal, M., Nie, J., Park, J., Luo, L., et al. High-performance, distributed training of large-scale deep learning recommendation models. *arXiv preprint arXiv:2104.05158*, 2021.

Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

Rendle, S. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pp. 995–1000. ieeexplore.ieee.org, December 2010.

Sharma, S. Feature fusion for the uninitiated | by siddharth sharma | medium. https://siddharth-1729-65206.medium.com/feature-fusion-for-the-uninitiated-4c5938db28b7, 2023. (Accessed on 01/24/2024).

Shin, K., Kwak, H., Kim, S. Y., Ramström, M. N., Jeong, J., Ha, J.-W., and Kim, K.-M. Scaling law for recommendation models: Towards general-purpose user representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 4596–4604, 2023.

Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., and Tang, J. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1161–1170, 2019.

Tang, J., Drori, Y., Chang, D., Sathiamoorthy, M., Gilmer, J., Wei, L., Yi, X., Hong, L., and Chi, E. H. Improving training stability for multitask ranking models in recommender systems. *arXiv preprint arXiv:2302.09178*, 2023.

Tianchi. Ad display/click data on taobao.com, 2018. URL https://tianchi.aliyun.com/dataset/dataDetail?dataId=56.

Tien, J.-B., joycenv, and Chapelle, O. Display advertising challenge, 2014. URL https://kaggle.com/competitions/criteo-display-ad-challenge.

Wang, R., Shivanna, R., Cheng, D., Jain, S., Lin, D., Hong, L., and Chi, E. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021a.

Wang, Z., She, Q., and Zhang, J. Masknet: Introducing feature-wise multiplication to ctr ranking models by instance-guided mask. *arXiv preprint arXiv:2102.07619*, 2021b.

Zha, D., Feng, L., Luo, L., Bhushanam, B., Liu, Z., Hu, Y., Nie, J., Huang, Y., Tian, Y., Kejariwal, A., et al. Pre-train and search: Efficient embedding table sharding with pre-trained neural cost models. *Proceedings of Machine Learning and Systems*, 5, 2023.

Zhang, G., Hou, Y., Lu, H., Chen, Y., Zhao, W. X., and Wen, J.-R. Scaling law of large sequential recommendation models. *arXiv preprint arXiv:2311.11351*, 2023.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023a.

Zhao, Z., Yang, Y., Wang, W., Liu, C., Shi, Y., Hu, W., Zhang, H., and Yang, S. Breaking the curse of quality saturation with user-centric ranking. *arXiv preprint arXiv:2305.15333*, 2023b.

Zhu, J., Liu, J., Yang, S., Zhang, Q., and He, X. Open benchmarking for click-through rate prediction. In Demartini, G., Zuccon, G., Culpepper, J. S., Huang, Z., and Tong, H. (eds.), *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, pp. 2759–2769. ACM, 2021.

doi: 10.1145/3459637.3482486. URL https://doi.org/10.1145/3459637.3482486.

Zhu, J., Dai, Q., Su, L., Ma, R., Liu, J., Cai, G., Xiao, X., and Zhang, R. BARS: towards open benchmarking for recommender systems. In Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Culpepper, J. S., and Kazai, G. (eds.), *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pp. 2912–2923. ACM, 2022a. doi: 10.1145/3477495.3531723. URL https://doi.org/10.1145/3477495.3531723.

Zhu, J., Dai, Q., Su, L., Ma, R., Liu, J., Cai, G., Xiao, X., and Zhang, R. BARS: towards open benchmarking for recommender systems. In Amigó, E., Castells, P., Gonzalo, J., Carterette, B., Culpepper, J. S., and Kazai, G. (eds.), *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, pp. 2912–2923. ACM, 2022b. doi: 10.1145/3477495.3531723. URL https://doi.org/10.1145/3477495.3531723.

A. Model-Specific Grid Search Space on Criteo

We use Adam for dense arch optimization and use Rowwise AdaGrad for sparse arch optimization with a linear warmup period for the first 10% steps. We use $8 * 16384 = 131,072$ global batch size. All models use ReLU for activation. We opted to use 128 as embedding dimension, as it shows better results on all models in our pilot experiments. We use FP32 in all runs. Due to the dataset volume and model size, we use (Mudigere et al., 2021) as the sparse distributed training framework and data parallel for dense synchronization.

To facilitate fair comparisons, we conducted extensive grid search (>3000 runs) over both general hyper-parameters and model-specific configs on Criteo Dataset.

For all the models, both sparse and dense learning rate was separately tuned in $\{1e^{-3}, 1e^{-2}, 1e^{-1}\}$. For MLPs in all the models, the number of hidden layers ranged in $\{1, 2, 3, 4\}$ with their layer sizes in $\{512, 1024, 2048\}$. To reduce the excessively large search space, we did a pilot experiments on the optimizer hyperparameters, and found that setting learning rate to $1e^{-3}$ for dense and $1e^{-1}$ for sparse works the best for all models. We fixed the learning rate in the following runs. We now describe model-specific search space:

AFN+ The AFN hidden units and DNN hidden units are the same across all runs, followed the general MLP search space. The number of logarithmic neurons ranges in $\{128, 256, 512, 1024\}$.

AutoInt+ We created the search space based on the best configurations reported in the paper (Song et al., 2019), with a larger value being considered additionally per hyperparameter. The number of attention layers ranged in $\{3, 4\}$, with attention dim ranged in $\{256, 512\}$. The number of attention heads are in $\{4, 8\}$. The DNN hidden units follow the general MLP search space.

DCNv2 The number of cross layers ranged from 1 to 4. Rank searched in either full-rank or 512.

DLRM The bottom MLP layer sizes and numbers was set to $[512, 256]$.

FinalMLP We followed the public benchmark setup (Zhu et al., 2022a), by setting FeatureSelection (FS) to all float features for one stream, and searching over one of 8 selected sparse features for the other stream. FS MLP is set to $[800]$. Number of heads is fixed to 256.

MaskNet We tested both Parallel MaskNet and Serial MaskNet. For the Parallel variant, we consider the number of blocks in $\{1, 8, 16\}$ and the block dimension in $\{64, 128\}$. For the Serial variant, we consider the number of layers in $\{1, 4, 8\}$ with the layer size in $\{64, 256, 1024\}$. We fixed the reduction ratio to 1 for both variants.

xDeepInt We considered Compressed Interaction Network (CIN) with the number of layers in $\{3, 4\}$ and the layer dimension in $\{16, 32, 64\}$.

Wukong The bottom MLP layer sizes and numbers was set to $[512, 256]$. l ranged from 1 to 4; n_F and n_L are set to the same value, ranged in $\{8, 16\}$. k is fixed to 24.

B. Model Complexity/Size on Public Datasets

Please refer to Table 3 for details.

C. Model-Specific Scaling-up Configurations

Please refer to Table 5 for details.

D. Analysis of High Order Interactions in Wukong

The traditional factorization machine approach solves second order interaction problem by minimizing (Naumov et al., 2019):

$$\min \sum_{i,j \in S} r_{ij} - X^1 {X^1}^T$$

where $r_{ij} \in R$ is the rating of the i-th product by the j-th user for i = 1, ..., m and j = 1, ..., n; X denotes the user and item representations (embeddings), and the superscript 1

	Frappe		MicroVideo		MovieLens	
	#Params	MFLOP	#Params	MFLOP	#Params	MFLOP
AFN+	13607961	81.45	3205771	10.57	11259899	33.36
AutoInt+	306151	15.98	1989378	10.74	1293590	3.00
DCNv2	7251073	39.35	3454209	11.84	1243101	2.02
DLRM	1527073	4.99	2853761	12.66	1238421	1.99
FinalMLP	3115954	10.66	3020498	9.25	1040902	0.81
MaskNet	21189593	122.92	2053249	7.87	2624571	10.30
xDeepFM	66206	0.01	5147446	5.91	994439	0.00
Wukong	7769589	44.09	2219409	9.08	17734369	37.49
	KuaiVideo		**TaobaoAds**		**Criteo**	
	#Params	MFLOP	#Params	MFLOP	#Params	MFLOP
AFN+	84013143	10.95	167888773	13.45	26436527675	1826.50
AutoInt+	43937794	79.18	167395330	28.41	26142619137	163.34
DCNv2	42636609	9.10	193472513	166.19	26179364097	262.35
DLRM	41446513	1.99	42192033	4.53	26136307201	4.62
FinalMLP	580227666	12.17	85624114	16.39	26149500924	84.95
MaskNet	41833034	4.29	42353201	5.44	26160209153	147.40
xDeepFM	53912381	6.81	168867028	2.98	26300221171	41.57
Wukong	44671649	22.27	175724173	63.00	26163636001	173.73

Table 3: Model complexity and size on public datasets.

denotes the embedding contains 1st order information. The dot product of these embedding vectors yields a meaningful prediction of the subsequent rating for 2nd order interactions. In Wukong, this meaningful interactions are then transformed to 2nd order interaction representations X^2 using MLP. In the 2nd layer FMB, with a residual and LCB connection, a dot product of $(X^1 + X^2)(X^1 + X^2)^T$ yield both meaningful interaction from 1st order to 4th order. By analogy, a l-layer Wukong solves a problem by minimizing:

$$\min \sum_{i,j \in S} (r_{ij} - \sum_{k \in 1,2,\ldots,2^{l-1}} X^k X^{k^T})$$

Thus, comparing to the traditional factorization approach, Wukong is able to solve the recommendation problem with a more sufficient interaction orders.

E. Scaling Law on Training Data Volume

Fig. 6 provides a summary for Wukong's performance versus the dataset size on which it is trained on (one pass). Similar to what has been observed on LLMs, we found that large models are more data-efficient, meaning that they require fewer samples to achieve the same quality improvement. In addition, we found that all Wukong models have consistently improved their model quality up to the end of 146B data, while larger models have a steeper trend in the model quality improvement. We also noticed that one of the limitations of our study is the dataset size is still far less for the large model to converge, which will be one of the areas for further study.

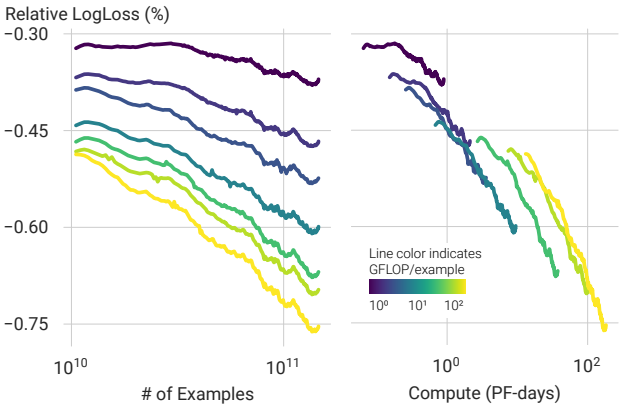

Figure 6: Wukong's model quality improvements versus training data volume and training compute.

F. Comparing with Transformer-based Approaches

We highlight differences and provide intuitions on why Wukong scales better than Transformer-based approaches like AutoInt+ (Song et al., 2019). While the structure of Wukong resembles that of Transformer's, we note the following architectural difference: first, the projection used in Wukong is MLP (bit-wise) in both FMB and each layer instead of FFN (embedding/position-wise) in transformer; second, Wukong is configured as a pyramid shape versus the uniform shape used in transformers.

We hypothesize that the difference in projection plays an important role in quality delivery. These MLPs operate over the flattened input embeddings, essentially providing each

feature with a different projection matrix. Our intuition is that this helps the model learn from heterogeneous input features, contrasting to a single embedding space used in LLMs. Similar intuition was discussed in (Gui et al., 2023).

Efficiency-wise, we argue that the pyramid shape configuration allows Wukong to exclude unnecessary computations by contracting the number of embeddings used in each layer.

To verify these hypothesis, we conduct the following experiments by applying Wukong's unique components to Autoint+, which conclude (1) using bit-wise MLP instead of FFN for V-projection improves LogLoss by 0.34%; (2) adding bit-wise MLPs after self-attentions improves LogLoss by 0.65%; (3) combining both, along with a pyramid layer shape (by using LCB on the first layer's output) achieves 0.57% quality improvement. Compared to the scaled up Autoint+, Wukong achieves 0.08% quality improvement, while saves 90% FLOPs. We summarize the results in Table 4

Changes vs. Autoint+	**Relative LogLoss** (%)	**GFLOP /example**
Vanilla Autoint+	0	8
V=FFN() $\rightarrow$ V=MLP()	-0.34	13
Scaled-up Autoint+	-0.49	50
V=FFN() $\rightarrow$ V=MLP() Layer FFN $\rightarrow$ Layer MLP Pyramid layer shape	-0.57	5
Layer FFN $\rightarrow$ Layer MLP	-0.65	36

Table 4: Replacing/Adding Wukong's unique components to vanilla Autoint+ improves the model quality.

Hyperparameters	GFLOP/example	#Params (B)	Relative LogLoss (Task1)	Relative LogLoss (Task2)
AFN+				
DNN=4x2048, afn=4x2048, nlog=1024	4.41	628.22	0.11	0.05
DNN=4x4096, afn=4x2048, nlog=1024	7.65	628.74	0.12	0.06
DNN=4x4096, afn=4x4096, nlog=2048	13.08	629.46	0.21	0.14
DNN=4x8192, afn=4x8192, nlog=4096	43.4	633.95	0.12	0.06
AutoInt+				
Attention=3x256, nhead=4, DNN=2x256	7.72	627.73	0.39	0.24
Attention=3x512, nhead=4, DNN=2x256	18.58	627.77	0.15	0.05
Attention=3x512, nhead=8, DNN=3x8192	42.53	631.49	-0.09	-0.16
Attention=3x512, nhead=16, DNN=3x10240	49.58	632.59	-0.1	-0.2
Attention=3x512, nhead=16, DNN=3x16384	68.83	635.57	0.13 (LossX)	0.01 (LossX)
DCN				
l=2, rank=512, MLP=4x2048	3	628.11	-0.27	-0.27
l=2, rank=512, MLP=4x4096	4.67	628.37	-0.29	-0.32
l=2, rank=512, MLP=4x16384	17.85	630.42	-0.38	-0.41
l=2, rank=512, MLP=4x32768	43.88	634.46	-0.43	-0.45
l=2, rank=512, MLP=4x51200	84.71	640.79	(LossX)	(LossX)
DLRM				
TopMLP=2x512	1.37	627.78	*(Baseline)*	*(Baseline)*
TopMLP=4x512	1.37	627.78	-0.11	-0.08
TopMLP=4x2048	3.85	628.17	-0.23	-0.21
TopMLP=4x4096	7.29	628.7	-0.28	-0.27
TopMLP=4x8192	14.61	629.84	-0.32	-0.31
TopMLP=4x16384	31	632.39	-0.37	-0.35
TopMLP=4x32768	71.23	638.62	-0.36	-0.34
FinalMLP				
MLP1=4x4096, MLP2=2x1024, output_dim=64, no_fs	3.93	628.25	-0.11	-0.16
MLP1=4x4096, MLP2=2x1024, output_dim=64, fs1=[0,57600], fs2=[57600,115200], fs_MLP=1x2048	8.17	628.91	-0.23	-0.27
MLP1=4x8192, MLP2=2x2048, output_dim=64, fs1=[0,57600], fs2=[57600,115200], fs_MLP=1x4096	16.9	630.27	-0.34	-0.36
MLP1=8x8192, MLP2=4x2048, output_dim=64, fs1=[0,57600], fs2=[57600,115200], fs_MLP=2x4096,	18.77	630.56	-0.37	-0.38
MLP1=4x16384, MLP2=2x4096, output_dim=64, fs1=[0,57600], fs2=[57600,115200], fs_MLP=1x8192	36.26	633.27	-0.34	-0.34
MLP1=4x24576, MLP2=2x6144, output_dim=64, fs1=[0,57600], fs2=[57600,115200], fs_MLP=1x12288	58.12	636.67	-0.37	-0.38
MaskNet				
MLP=1x512, nblock=1, dim=128, reduction=0.01	1.76	627.92	-0.09	-0.12
MLP=1x512, nblock=4, dim=128, reduction=0.01	6.8	628.7	-0.22	-0.25
MLP=3x2048, nblock=4, dim=128, reduction=0.01	6.88	628.71	-0.28	-0.3
MLP=3x2048, nblock=4, dim=128, reduction=0.05	32.36	632.67	-0.37	-0.37
MLP=3x2048, nblock=4, dim=128, reduction=0.1	64.21	637.61	-0.4	-0.4
Wukong				
l=2, nL=8, nF=8, k=24, MLP=3x2048	0.53	627.74	-0.35	-0.32
l=4, nL=32, nF=32, k=24, MLP=3x2048	1.25	627.82	-0.45	-0.43
l=8, nL=32, nF=32, k=24, MLP=3x2048	2.12	627.95	-0.53	-0.49
l=8, nL=48, nF=48, k=48, MLP=3x4096	5.6	628.46	-0.6	-0.6
l=8, nL=96, nF=96, k=96, MLP=3x8192	22.23	630.96	-0.67	-0.66
l=8, nL=96, nF=96, k=96, MLP=3x16384	61	636.99	-0.7	-0.69
l=8, nL=192, nF=192, k=192, MLP=3x16384	108	644	-0.76	-0.76

Table 5: Detailed hyperparameters, compute complexity, model quality and model size for each run evaluated in Sec. 7. LossX means loss exploded during training.