MAGLEV: IMPROVING THE RECOMMENDATION SCALING LAWS VIA UNIFIED SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Industrial recommenders face diminishing returns from dense scaling, while sparse and depth scaling make global feature interaction and distributed training prohibitively expensive. We introduce Maglev, a model–system co-design that decomposes a recommender into hardware-local pipeline stages. It combines (1) Maglev Indexers, which learn capacity-bounded stage inputs, as soft projections within one scale-up domain or hard feature partitions across them; (2) Maglev Induct, which carries a low-rank residual across stages; and (3) Maglev Rail, a pipeline schedule that stays bubble-free with synchronous gradient clipping. We conduct an industrial-scale study that jointly scales sparse capacity and dense compute (28B samples, 700 features, four tasks). Applied to Wukong and DHEN, the strongest of 10 screened backbones, Maglev improves the scaling law along every axis: every Maglev configuration outperforms the best baseline configuration at equal or lower compute and sparse capacity, and the lead widens as sparse capacity and compute grow. The full system trains up to $1.3\times$ faster than hybrid parallelism. On three public benchmarks, Maglev improves five further backbones in all 15 matched-compute settings. Maglev has been launched in three flagship recommenders at a major social media company, improving RLL by up to 0.4% in A/B tests.

1 INTRODUCTION

Industrial recommenders now operate at LLM-scale training complexity (Luo et al., 2025; Zhai et al., 2024; Liu et al., 2026), yet quality returns from dense scaling are diminishing sharply: under the recommendation scaling laws (Zhang et al., 2024a; Xu et al., 2025a; Zhang et al., 2024b; Shin et al., 2023; Yan et al., 2026; Guo et al., 2024), every constant gain in relative log-loss now demands exponentially more compute. This trajectory is incompatible with both continuous online training (Gao et al., 2025) and strict serving-latency budgets, leaving dense-only scaling increasingly less attractive for sustainable quality growth.

The natural alternatives, such as *sparse scaling* (more features, longer histories, larger embeddings (Luo et al., 2025; DeepSeek-AI et al., 2025)) and *depth scaling*[1] (more interaction layers under a fixed FLOPs budget), have stalled for a single architectural reason. Today's monolithic recommenders perform every embedding lookup up-front and let every feature interact with every other, producing two compounding costs: quadratic compute in feature count, and all-to-all embedding traffic that exceeds what datacenter scale-up domains can deliver (Luo et al., 2024; 2018; Zhang et al., 2025; Gholami et al., 2024). Depth also saturates rapidly in narrow monolithic stacks (Zhang et al., 2024a; 2023), mirroring observations in LLMs (Hoffmann et al., 2022; Kaplan et al., 2020; Petty et al., 2023). We diagnose these walls empirically in §2.

We propose Maglev, a model–system co-design built around the *Maglev Stage*: a localized unit consisting of a feature partition and an interaction module, mapped to a single hardware scale-up domain (HSD). An end-to-end Maglev model is a sequence of such stages connected by pipeline parallelism (Harlap et al., 2018). This single primitive reduces sparse, depth, and dense scaling to *widening a stage* or *adding a stage*. Maglev makes this possible through three components. **Maglev Indexers** select each stage's inputs under a fixed output budget, while **Maglev Induct** carries an

[1]Not to be confused with *weak depth scaling*, which adds layers without holding total FLOPs fixed; we treat weak depth scaling as a form of dense scaling.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

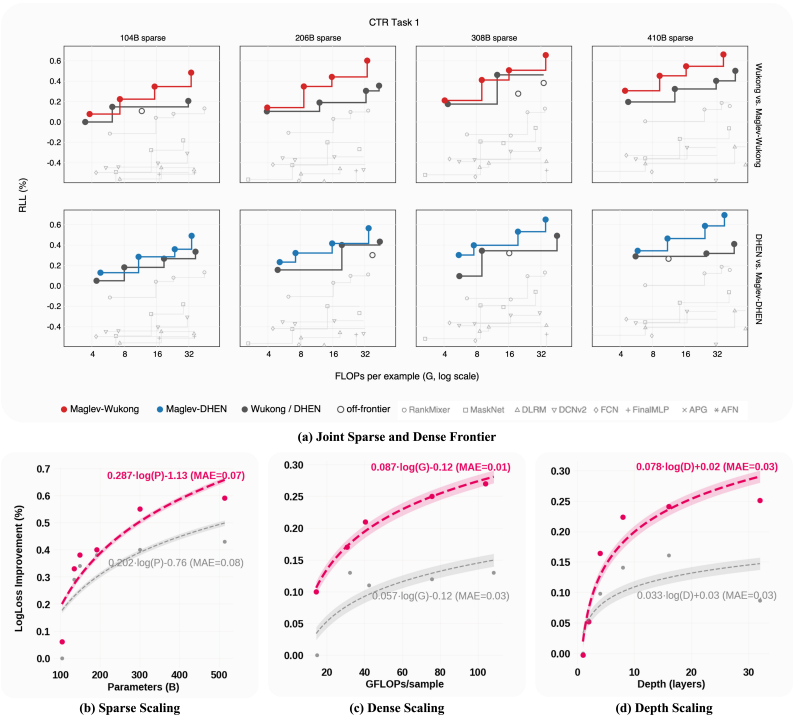

Figure 1: Maglev improves the recommendation scaling curves. (a) Quality–compute frontiers on CTR Task 1: Maglev-Wukong (red) vs. Wukong and Maglev-DHEN (blue) vs. DHEN (dark gray; rows), at each sparse capacity (columns). Each step line traces the best RLL reachable at equal or lower FLOPs; hollow markers are configurations below their arm's frontier. Maglev's frontier lies above the baseline's at every compute and sparse capacity, and the gap widens as sparse capacity grows. Light gray: the other eight families (configurations below the plotted range omitted). (b)–(d) Isolated sparse, dense and depth sweeps, Maglev-Wukong (highlighted) vs. Wukong (gray).

inter-stage residual that causally conditions each stage on prior outputs. Together, they replace global pairwise interaction with capacity-bounded stage-local computation. For a model that fits in one HSD, these two components are the whole method: Soft Indexers learn each stage's inputs as projections of all features, with no partitioning step or special schedule. Once a model spans several HSDs, Hard Indexers turn the same learned affinities into a feature partition, so each stage reads only its own embedding shard, allowing efficient pipelined execution with **Maglev Rail**. This makes exact synchronous gradient clipping bubble-free without speculative rollback possible, at 1F1B's memory and communication cost (under the standard equal-latency model).

We jointly scale sparse capacity and dense compute at industrial scale. We validate Maglev on a 28-billion-sample, 700-feature dataset spanning four tasks. We apply Maglev to Wukong and DHEN, the strongest of 10 architecture families in a shared-protocol screen. For reproducible external validation, we additionally transform five public backbones on KKBox, Criteo, and KuaiVideo.

Findings Maglev lifts the quality–compute frontier of both backbones on all four tasks. Every Maglev configuration outperforms the best baseline configuration at equal or lower FLOPs and sparse capacity; for 70 of 128 configuration–task pairs, no configuration of the same backbone reaches the same quality at any compute we trained. In isolated Wukong sweeps, the baseline flattens beyond $\sim$40 GFLOPs (dense) and $\sim$300B parameters (sparse) and degrades past 16 layers (depth), whereas Maglev continues to improve along dense and sparse axes and retains its gains through 32 stages. The full system trains up to $1.3\times$ faster.

On public benchmarks, Maglev improves all 15 backbone–dataset settings at matched compute (14 of 15 paired 95% intervals exclude zero). Three flagship Maglev recommenders have been launched at a major social media company with up to 0.4% RLL improvement.

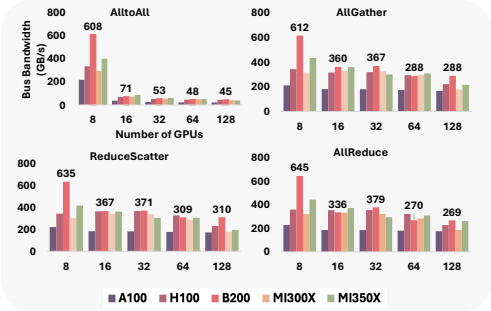

Figure 2: Collectives performance vs GPU count across hardware generations and vendors.

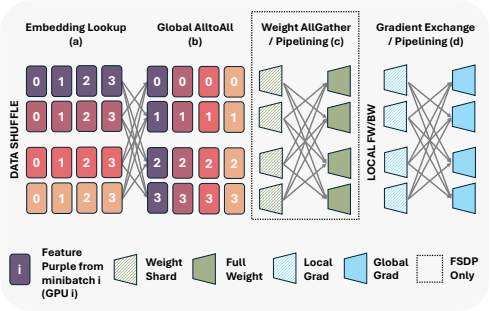

Figure 3: The SOTA training paradigm of a 4-feature recommender on 4 GPUs.

2 Scaling Challenges and Opportunities

The Dense Scaling Wall We assess dense scaling efficiency on a Wukong model (Zhang et al., 2024a), which is the architecture that established the baseline recommendation scaling laws, by varying its compute budget from 15 to 110 GFLOPs per sample. As Figure 1(c, gray) shows, relative log-loss (RLL) improvements plateau beyond $\sim$40 GFLOPs/sample. Dense scaling alone yields strictly diminishing returns, motivating the alternative axes below.

The Sparse Scaling Wall Sparse capacity is expanded along three axes: (1) adding feature fields through multi-domain, multi-objective, and surface consolidation (Li & Tuzhilin, 2020; Sheng et al., 2021; Wang et al., 2022; Luo et al., 2025; Feng et al., 2026); (2) lengthening user histories (DeepSeek-AI et al., 2025); and (3) widening embeddings. Each axis magnifies the same architectural cost: the feature-interaction modules at the heart of industrial recommenders are super-linear in either feature count F or flattened embedding size FD.

- **Explicit Factorization Machines** (Naumov et al., 2019; Zhang et al., 2024a): pairwise dot products $\langle f_i, f_j \rangle$ produce an $F \times F$ score matrix at $\mathcal{O}(F^2 D)$ cost (Naumov et al., 2019).
- **Attention** (Vaswani et al., 2017; Song et al., 2019): pairwise attention over the length-F embedding sequence, also $\mathcal{O}(F^2 D)$.
- **Cross Networks** (Wang et al., 2021a): a per-layer dense weight $W \in \mathbb{R}^{FD \times FD}$ on the flattened embedding, costing $\mathcal{O}((FD)^2)$ per layer.

This super-linearity is what makes sparse scaling unprofitable in practice. Adding features inflates interaction cost quadratically; to stay within the serving-latency budget, production systems must shrink dense hyperparameters (narrower MLPs, fewer interaction layers) or fall back to distillation (Luo et al., 2025; Liang et al., 2025), which results in subpar efficacy (Liu et al., 2026). Figure 1(b, gray) shows the empirical signature: RLL plateaus past 300B sparse parameters under iso-FLOP, iso-depth conditions.

The Depth Scaling Wall Under a fixed FLOPs budget, strong depth scaling is uniquely attractive: it grows neither parameters nor compute, yet stacked interaction layers can in principle capture progressively higher-order feature interactions (Zhang et al., 2024a) and improve generalization. In practice, however, depth saturates rapidly in monolithic recommenders. Scaling a 6-GFLOPs Wukong from 1 to 32 layers traces a U-shape in RLL (Figure 1(d)): deeper architectures must sacrifice width, and beyond 16 layers the trade-off degrades quality outright — mirroring saturation observed in LLMs (Hoffmann et al., 2022; Kaplan et al., 2020; Petty et al., 2023).

The Training Paradigm Mismatch SOTA recommenders execute via hybrid parallelism (Mudigere et al., 2021; Ivchenko et al., 2022): model-parallel embedding exchange for sparse activations interleaved with pipelining and data-parallel (Zhao et al., 2023) gradient synchronization for dense parameters (Figure 3). The resulting all-to-all traffic stresses datacenter networks, whose throughput collapses once they span more than a single hardware scale-up domain (HSD, e.g., a server) (Si et al., 2026; Luo et al., 2018) (Figure 2). Despite balancing (Zha et al., 2023) and locality-aware reduction (Luo et al., 2024; 2017; 2020), scaling beyond a single HSD often forces asynchronous training (Zhang et al., 2025), which directly harms convergence.

3 MAGLEV

A Maglev model is a sequence of *Maglev Stages*, each consisting of a feature partition and an interaction module mapped to a single hardware scale-up domain (HSD). Stages are connected via pipeline parallelism (Harlap et al., 2018). This decomposition collapses the upfront global feature interaction and the quadratic cost it incurs into fixed, per-stage local computation. Naive chunking, however, fragments feature interactions, accrues pipeline bubbles, and starves cross-stage information bandwidth. The three components resolve these failure modes and attack the walls of §2. Maglev Indexers defines the model and suffice when it fits in one HSD; Hard Indexers, Maglev Induct and Maglev Rail are needed once stages span several HSDs.

3.1 MAGLEV INDEXERS

Soft Maglev Indexers The Soft Indexer is the complete form for models that fit in one HSD and the pretraining form for Hard Indexers. It takes a global set of F features with dimension D ($\mathcal{F} \in \mathbb{R}^{F\times D}$) and an output size budget P_i for each Maglev stage $i \in [1 \dots K]$. It outputs a P_i-sized set of linear combinations of the global features, denoted Y_i. These combinations are parameterized by a learnable *feature-stage affinity matrix* $\mathcal{A} \in \mathbb{R}^{F\times\sum_{m=1}^{K} P_m}$, where $\mathcal{A}_{j\ell}$ is the weight of feature j in global output combination ℓ. The affinity matrix $\mathcal{A}$ is jointly optimized during standard training. Operationally, we compute a globally fused output $Y_{\text{fused}} = \mathcal{A}^\top \mathcal{F}$ and cleanly slice the partition for the i-th stage:

$$Y_i = Y_{\text{fused}} \left[\sum_{m=1}^{i-1} P_m : \sum_{m=1}^{i} P_m, : \right]$$

Hard Maglev Indexers To avoid global all-to-all communications or holding the entire global feature embeddings, linear combinations must be confined strictly to features within the local shard. Hard Maglev Indexers derives an affinity-driven sparse sharding strategy from $\mathcal{A}$ in two steps. First, we pretrain a proxy model to establish $\mathcal{A}$, a one-time cost per model family. To make this pretrain run feasible, we use a model with shrunk embedding sizes and dense dimensions, but keeping the model architecture intact[2]. Second, let S_i denote stage i's raw feature-field storage budget and P_i its compressed output width. Using the stage slice $\mathcal{A}_i \in \mathbb{R}^{F\times P_i}$, we score each field by its mean absolute affinity and select S_i fields:

$$v_i = \frac{1}{P_i} \sum_{k=1}^{P_i} |\mathcal{A}_i[:, k]| \in \mathbb{R}^F, \qquad \mathcal{F}_i = \text{TopK}_{S_i}(v_i).\,\text{Indices}\,.$$

For every field assigned to $\mathcal{F}_i$, stage i owns a stage-local embedding-table instance. Let $\mathcal{E}_i \in \mathbb{R}^{S_i\times D}$ denote its lookup outputs; the stage applies the learned local compression

$$Y_i = W_i^\top \mathcal{E}_i \in \mathbb{R}^{P_i\times D}, \qquad W_i \in \mathbb{R}^{S_i\times P_i}.$$

Thus, S_i controls feature placement and storage, whereas P_i controls output bandwidth. Because independent top-S_i selections may omit long-tail fields, we require $\sum_i S_i \geq F$ and use the greedy procedure in Appendix A.1 to replace lowest-affinity duplicate assignments with missing fields while preserving every stage's S_i budget. All Maglev experiments on our industrial dataset use $\sum_i S_i = F$: each field, however wide its embedding, is assigned to exactly one stage. Field reuse, which places a field in several stages, is studied only on a separate dataset (§A.12.2).

Indexer Costs. With Maglev Indexers, the global $\mathcal{O}(F^2D)$ pairwise interaction is replaced by the stage-local interaction term $\mathcal{O}(P_i^2 D)$. Hard-Indexer compression additionally costs $\mathcal{O}(S_i P_i D)$ at stage i, giving the aggregate deployed cost

$$\mathcal{O}\left(D \sum_{i=1}^{K} (S_i P_i + P_i^2) \right).$$

For fixed S_i and P_i, each stage's cost is independent of the global feature count F. Under balanced storage $S_i \approx F/K$ and the empirical heuristic $P_i \approx F/(2K)$, the compression and interaction terms scale as $F^2D/(2K)$ and $F^2D/(4K)$, respectively, retaining a $\Theta(K)$ reduction from global

[2] In Appendix §A.5, we show Maglev Indexers stabilizes quickly after warmup, making this pretraining run efficient – taking only hours instead of weeks compared to full runs.

interaction. The Soft Indexer keeps the global projection $\mathcal{O}(FD\sum_i P_i)$, a single dense GEMM that grows linearly in F at fixed output budgets, and it needs every feature at every stage. Within one HSD both costs are small, so the Soft Indexer is the deployed form there. Hard Indexers remove the remaining dependence on F and the cross-HSD embedding exchange once stages span several HSDs.

Fused Hard-Indexer kernel. Because hard assignments are static, one kernel streams the selected embeddings directly into grouped tensor-core GEMMs, without separate gather and concatenation operators; on B200 it is 2.3× faster than naive `torch.compile`.

3.2 MAGLEV INDUCT

Maglev Induct targets both the Depth and Dense Walls and the residual information loss left by Maglev Indexers, by enforcing causal, conditional learning of the current stage's representation on prior stage's output. Hard Maglev Indexers reduces dense pairwise interactions to sparse, block-diagonal ones (Ye et al., 2024; Ma et al., 2024): even when the most important features are co-located, the long tail of feature interactions ends up split across stages and cannot be computed locally. This lost long-tail context can degrade quality even when the head interactions are preserved (Tay et al., 2020; Xiao et al., 2024). Maglev Induct recovers it by carrying a low-rank residual R across stages. *Where traditional recommenders fix every feature's effective representation upfront in the embedding table, in Maglev the effective representation of stage i's features is dynamically conditioned on what prior stages extracted*; adding depth in Maglev therefore means adding new conditional context and increasing residual bandwidth (Xie et al., 2026; Zhu et al., 2025a), leading to simultaneously better depth and dense scaling efficiency.

Specifically, Maglev Induct ingests three inputs: the dense activations from the previous stage (X), the accumulated residual feature digest (R), and the linear-compressed indexer output for the current stage (Y). The memory R, carrying historical feature context, is fed directly into the current stage's dense layer to recover cross-stage interactions. A newly updated digest R' is then produced by incorporating the current stage's signals and passed to the next stage via a gated residual summation, $R' = \mathrm{LN}\big(\lambda R + \mu\,(YW_y \odot \sigma(XW_x))\big)$, where λ and μ are learned scalars, W_y and W_x project Y and X to the width of R, σ is the sigmoid function, and $\odot$ is the Hadamard product. The gate follows gated attention (Qiu et al., 2025) and conditional representation (Yu et al., 2025).

Anatomy of a stage. Stage i receives the previous stage's dense activations X, its Indexer output $Y = Y_i \in \mathbb{R}^{P_i \times D}$ compressed from its S_i locally owned fields, and the accumulated digest $R \in \mathbb{R}^{O \times D}$, where O is the Maglev Induct width. These are concatenated and fed to one unmodified backbone block, and the updated digest R' is passed to stage $i+1$. A K-stage Maglev-Wukong therefore has the depth of a K-layer Wukong, but each layer interacts over $|X| + P_i + O$ tokens instead of all F features. Dimension mismatch is handled with a linear projection on the feature axis.

Linear-Surrogate Representation Bound To size the bottleneck Maglev Induct introduces, we analyze an additive linear surrogate. Let $Z = \mathcal{F}\mathcal{F}^\top \in \mathbb{R}^{F \times F}$ denote the global pairwise feature interactions, let $h_\star = ZW$ denote the monolithic readout, and let $B_i \in \mathbb{R}^{F \times O}$ denote the accumulated affinity factor available at stage i, which maps features into the width-O digest. The corresponding surrogate output is $\widehat{h}_i = ZB_iU_i$ with readout U_i.

For Hard Maglev Indexers, let Q_i select the feature fields available through stage i, and define $\Pi_i = B_iB_i^\dagger$. With the optimal linear readout $U_i^\star = B_i^\dagger W$,

$$\left\| h_\star - \widehat{h}_i \right\|_2^2 \le \|z(\mathbf{f})\|_2^2 \left(\underbrace{\|(I - Q_i)W\|_F^2}_{\varepsilon^{(i)}_{\mathrm{defer}}} + \underbrace{\|(Q_i - \Pi_i)W\|_F^2}_{\varepsilon^{(i)}_{\mathrm{rank}}} \right). \tag{1}$$

The first term captures interactions deferred to future stages; the second captures width-O truncation and affinity-subspace mismatch. Full feature coverage gives $Q_K = I$, eliminating the deferral term at the final stage, while the remaining error is lower-bounded by the rank-O spectral tail $\sum_{k>O} \sigma_k^2(W)$. The complete derivation and its Lipschitz transfer to the deployed gated block appear in Appendix A.9. The bound makes disaggregation a tunable trade-off. Coverage ($\sum_i S_i \ge F$) removes $\varepsilon_{\mathrm{defer}}$ by the final stage, leaving a rank-O floor that depends on neither the partition nor K. In the surrogate, O is then the only design choice that sets how much cross-stage interaction is lost, so stages can be

split for locality and the saved interaction FLOPs spent on depth or dense width. The deployed gate contains this ungated update as the case $W_x = 0$ and makes the retained subspace input-dependent, so at its optimum it cannot do worse (§ A.9).

3.3 MAGLEV RAIL

Maglev Rail binds each Maglev stage to one hardware scale-up domain (HSD), such as an NVLink domain (nvl). A stage's embedding tables, derived from the Maglev Indexers partition, are sharded only within its HSD, which removes the global all-to-all embedding exchange from the critical path; activations flow peer-to-peer between consecutive stages. Equal partition sizes, matched block dimensions, and a fixed Induct width keep stages balanced.

Maglev Rail builds on the zero-bubble schedules of Qi et al. (2023). It keeps the rectangular iteration of ZB-H1, which meets a synchronous clipping barrier without a drain, and fills ZB-H1's warm-up idle slots with each stage's embedding lookups. They can go there because lookups have no cross-stage dependency. With one microbatch per stage ($M = K$), the schedule becomes a rectangle: every stage finishes its weight gradients at the same step, so exact synchronous gradient clipping adds no drain, while activation memory and boundary communication match 1F1B (proof under the standard equal-latency model in Appendix A.2).

Figure 5 shows the resulting schedule. We also overlap communication with compute through several low-level optimizations, including Pipeline Parallel Table-wise Column-wise Sharding (PPTwCw), an intra-HSD hierarchical sharding strategy (Luo et al., 2024) (Appendix A.3).

4 EXPERIMENTS

4.1 SETUP

Datasets and baselines Our primary study uses a proprietary dataset containing 28B samples, 700 feature fields (about 600 sparse and 100 sequence), and four tasks. An RLL improvement of 0.01% is considered practically significant (§A.10). Wukong and DHEN, the strongest of 10 architecture families in a shared-protocol screen (§ A.6.2), are selected for Maglev transformation. For reproducible external validation, we additionally evaluate five public backbones on KKBox (kkb), Criteo (Dis), and KuaiVideo (Kuaishou). Details are in §A.10.

Metrics and uncertainty We report relative log-loss, $\mathrm{RLL}(X) = 100\%[\mathrm{LogLoss}(\mathrm{base}) - \mathrm{LogLoss}(X)]/\mathrm{LogLoss}(\mathrm{base})$, where higher is better. Each industrial configuration is trained once (the sweep used ∼800K H100-normalized GPU-hours). A five-seed, two-hardware repeatability check gives a seed standard deviation of at most 0.013 pp on the reported deltas (§A.10.1), well below the effects we report. Public benchmarks use 12 paired seeds with 95% confidence intervals.

4.2 PUBLIC DATASETS

We evaluate RankMixer (Zhu et al., 2025b), FinalMLP (Mao et al., 2023), AutoInt (Song et al., 2019), DESTINE (Xu et al., 2021), and FiGNN (Li et al., 2019) with their Maglev counterparts on KKBox, Criteo, and KuaiVideo. Every setting uses validation-only selection under equal tuning budgets, 12 matched held-out seeds, and FLOPs matched within 1.09% and all trainable parameters within 4.8% (Maglev never exceeds native by more than 1.1%). We report AUC on KKBox and Criteo and gAUC on KuaiVideo, with paired Student-t intervals.

Across five backbones and three datasets, Maglev produces a positive mean delta in all 15 settings; 14 of 15 nominal paired 95% intervals exclude zero, and the same 14 settings remain after Holm correction across the 15 paired tests (Table 1). Selected configurations appear in § A.6.1.

4.3 SCALING LAWS

We conduct an industrial-scale study to jointly scale sparse capacity and dense compute for recommendation models, then isolate each axis.

Multi-dimensional Scaling Within the industrial campaign, we compare 64 Wukong, DHEN, and corresponding Maglev configurations, four at each of four sparse capacities (P, 104–410B, scaled by widening every field's embedding 1–4×), and evaluate each on four tasks. Because the arms'

Table 1: Public dataset results. L/K: depth/Maglev stage count; p: Indexer output width per stage; $\Delta = m^{+\text{Maglev}} - m^{\text{native}}$. Every setting uses $n = 12$ matched seeds; N/M denotes native/+Maglev. MFLOPs and parameter counts (millions) are rounded to two decimals. Intervals are nominal paired 95% CIs. KKBox and Criteo report AUC; KuaiVideo reports gAUC.

Backbone	Dataset	L/K	p	Native	+Maglev	Δ [95% CI]	MFLOPs N/M	Params (M) N/M
RankMixer	KKBox	3/3	13	0.815562	**0.818318**	+0.002756 [0.001962,0.003550]	0.50/0.50	1.70/1.70
	Criteo	9/9	72	0.810631	**0.811749**	+0.001117 [0.000904,0.001331]	5.00/5.00	15.83/15.60
	KuaiVideo	9/9	11	0.648478	**0.649948**	+0.001470 [0.000694,0.002245]	4.98/4.98	41.33/41.29
FinalMLP	KKBox	6/6	7	0.813938	**0.814895**	+0.000957 [0.000286,0.001628]	3.77/3.77	2.81/2.79
	Criteo	2/2	13	0.806013	**0.806320**	+0.000307 [0.000164,0.000450]	0.41/0.41	8.58/8.57
	KuaiVideo	3/3	4	0.635644	**0.639249**	+0.003604 [0.002081,0.005128]	2.46/2.46	40.15/40.13
AutoInt	KKBox	2/2	8	0.810655	**0.813334**	+0.002679 [0.001511,0.003847]	0.30/0.30	0.93/0.94
	Criteo	2/2	34	0.802731	**0.804484**	+0.001754 [0.001614,0.001894]	1.30/1.30	8.39/8.40
	KuaiVideo	3/3	3	0.632544	**0.633641**	+0.001097 [-0.000025,0.002219]	0.56/0.56	38.95/39.01
DESTINE	KKBox	3/3	5	0.804340	**0.812475**	+0.008135 [0.004543,0.011727]	0.18/0.18	1.49/1.50
	Criteo	3/3	13	0.803869	**0.804427**	+0.000557 [0.000359,0.000756]	1.22/1.22	13.46/13.49
	KuaiVideo	3/3	3	0.626969	**0.632760**	+0.005792 [0.002490,0.009093]	0.92/0.92	39.01/39.09
FiGNN	KKBox	4/4	13	0.808652	**0.816733**	+0.008080 [0.006297,0.009864]	1.47/1.46	3.85/3.76
	Criteo	2/2	39	0.803982	**0.804934**	+0.000952 [0.000369,0.001535]	0.91/0.91	17.70/16.86
	KuaiVideo	2/2	7	0.628795	**0.633893**	+0.005098 [0.003722,0.006474]	0.97/0.98	39.06/39.06

Table 2: Maglev lifts the quality–compute frontier on every task and backbone. Each Maglev configuration is compared with the same-backbone baseline frontier at the same sparse capacity: the best baseline RLL at equal or lower FLOPs, interpolated in $\log G$ to the Maglev configuration's FLOPs. *Wins*: comparisons where Maglev is above the frontier; three Maglev configurations cheaper than every baseline at their capacity have no comparator. *Margin*: median RLL lead (pp). *Compute*: median baseline FLOPs needed to reach Maglev's RLL, relative to Maglev's, over the n configurations some baseline reaches. *Unreached*: Maglev configurations that no baseline configuration at the same capacity reaches at any trained compute.

	Task	Backbone	Wins	Margin (pp)	Compute (n)	Unreached
CTR	Task 1	Wukong	14/14	0.18	1.6× (7)	9/16
		DHEN	15/15	0.12	1.8× (8)	8/16
CVR	Task 1	Wukong	14/14	0.29	1.3× (5)	11/16
		DHEN	15/15	0.15	1.6× (7)	9/16
	Task 2	Wukong	14/14	0.16	1.9× (8)	8/16
		DHEN	15/15	0.16	1.5× (8)	8/16
	Task 3	Wukong	14/14	0.18	1.8× (7)	9/16
		DHEN	15/15	0.12	1.6× (8)	8/16
All			**116/116**	**0.15**	**1.6**× (58)	**70/128**

compute grids do not align exactly, we compare each Maglev configuration with the same-backbone baseline frontier: the best RLL at the same P and equal or lower FLOPs (G), interpolated to the Maglev configuration's FLOPs (§A.11). Maglev lies above this frontier in all 116 comparisons (Table 2, Figure 1(a)), with a median margin of 0.15 pp that grows from 0.10 pp at 104B to 0.21 pp at 410B. Where a same-backbone baseline configuration reaches a Maglev configuration's quality, it needs 1.1–$5.8\times$ the compute (median $1.6\times$); for 70 of 128 pairs, none we trained reaches it at all. Maglev also lies above the combined frontier of all 10 families in all 128 configuration–task pairs.

The joint grid conflates sparse and dense scaling, because in non-Maglev architectures, growing sparse capacity inflates FLOPs. We isolate each axis below, benchmarking Maglev-Wukong against the standard Wukong baseline that established the original recommendation scaling laws.

Sparse Scaling At fixed depth (10) and ≈30 GFLOPs, we sweep sparse capacity from 104B to 514B parameters, first by adding features and then by widening embeddings (§A.6.5). Wukong's first layer interacts over every input token, and both steps add tokens, so holding it at 30–32 GFLOPs means narrowing its MLP 7×, from 3648-5472 at 104B to 512-768 at 514B. Each Maglev stage interacts only over its own partition, so its MLP keeps its full width (4112-6168) through 191B and narrows by only 20% by 514B, at flat compute (30.69 vs. 30.63 GFLOPs); its Indexer output is 64 up to 149B and 96 from 191B.

Table 3: **Comparison with DMT at 104B sparse capacity.** All configurations use directly comparable data, optimization, and training protocols.

Configuration	Structure	GFLOPs	Partition Strategy	ΔRLL
Wukong	4 layers	3.47	-	0.00%
Wukong	8 layers	6.20	-	+0.14%
DMT-Wukong	8 partitions	5.10	Coherent	−0.15%
DMT-Wukong	16 partitions	6.40	Coherent	−0.26%
Maglev-Wukong	4 stages	3.81	Indexer	**+0.08%**
Maglev-Wukong	8 stages	6.57	Indexer	**+0.22%**

Dense Scaling At fixed depth (10) and sparse capacity (135B), we sweep dense compute per sample, widening the top MLP per Wukong's scaling guide and co-tuning other internal widths at each budget. Figure 1(c) shows Maglev-Wukong breaching the dense wall reached by the baseline at approximately 40 GFLOPs, achieving better absolute performance and a $1.5\times$ larger fitted coefficient over the evaluated range. The joint grid, which varies depth, LCE, FM, and Rank rather than following one widening recipe, gives the same ordering: Maglev lies above the baseline frontier at every compute (Table 2).

Depth Scaling At fixed sparse capacity (104B) and near-matched compute (Maglev: 5.50–6.57 GFLOPs; Wukong: 6.12–6.39 GFLOPs), we vary model depth from 1 to 32 layers/stages. Both arms narrow as they deepen to hold compute (§A.6.3), so this sweep compares depth–width allocations at fixed compute rather than depth alone. Figure 1(d) reproduces the U-shape of standard Wukong, with sharp degradation past 16 layers as monolithic stacks become too thin; Maglev ties Wukong at one to two stages, outperforms it from four stages onward, and remains non-decreasing through 32 stages. Gains taper to $+0.01\%$ RLL from 16 to 32 stages, while Wukong falls back to 0.09%; the baseline's non-monotone curve makes the fitted $2.4\times$ ratio a summary of the observed range rather than the primary evidence.

4.4 Production Deployment

Maglev targets quality within a fixed serving budget: Maglev Rail is used only in training, the model is exported as one conventional serving graph, and Indexer and Induct compute are counted in the matched budget. At a major social media company, three independently trained flagship recommenders using Maglev have been launched in production. Before launch, each model underwent a 14-day randomized A/B test against its incumbent production baseline, covering two CVR models and one joint CTR+CVR model. All three met their pre-established launch criteria and improved RLL by $+0.1\%$, $+0.3\%$, and $+0.4\%$.

4.5 Ablation

Figure 4 isolates the contributions of Maglev's components under the fixed-seed industrial protocol. Detailed protocols and additional analysis appear in § A.12.

Indexers and Induct Naive feature disaggregation reduces RLL by 0.02–0.05%. The Soft Indexer recovers the loss and improves RLL by 0.04%; the deployable Hard Indexer alone reaches $+0.01\%$, and adding Maglev Induct raises it to $+0.05\%$, so Maglev Induct recovers what locality costs. On a 10-stage production model, the Hard-Indexer model beats a Soft-Indexer model without Maglev Induct by 0.07 pp at 6% fewer FLOPs (§A.12.2). Learned partitioning outperforms random and importance-based partitioning, with diminishing returns beyond width 64 (+0.04 from 32→64, +0.02 from 64→128). For Maglev Induct, $O = 32$ reduces interaction complexity by 14% with negligible quality loss, while gating adds 0.03% RLL. Affinity assignments also stabilize early (§ A.5).

DMT Comparison We compare with DMT (Luo et al., 2024), a KMeans-based partition scheme. DMT uses its coherent partitioning strategy, a tower compression width equal to Maglev's Indexer width, and Wukong's LCE, FM, and Rank, with the MLP tuned to match FLOPs where the architecture allows. Both inherit Wukong's native residuals, but DMT splits all features into independent towers at the first layer and fuses them only at the end, so interactions a tower misses cannot be recovered; we attribute DMT's drop below the 4-layer baseline to this. Maglev's sequential stages carry such context forward through Maglev Induct. At near-matched compute, Maglev achieves $+0.22\%$ RLL versus DMT's -0.26%—a 0.48 percentage-point gap at 2.7% higher FLOPs; Figure 4 separately isolates Maglev Induct.



Figure 4: Ablation of the Maglev components under the fixed-seed industrial protocol.

Sparse scaling Horizontal scaling performs best, while widening existing embeddings provides gains without additional feature engineering (Figure 4(d)).

Maglev Rail throughput Relative to the original training system running hybrid parallelism (ZB2P+FSDP), the complete system delivers up to $1.3\times$ end-to-end throughput, with the component breakdown reported in § A.12.5. Separately, the isolated schedule improves throughput over ZB2P by 6.3%–11.7% across all 12 measured configurations (§ A.4).

5 Related Work and Discussion

Prior recommendation scaling studies either examine dense or sparse axes in isolation (Zhang et al., 2024a; Hou et al., 2026; Zeng et al., 2024; Xu et al., 2025b; Shin et al., 2023) or jointly fit sparse and dense parameters of an existing architecture (Ardalani et al., 2022). Maglev instead redesigns interaction so that quality keeps rising with sparse capacity, and scales it jointly with dense compute and depth, and co-designs the distributed execution. Unlike token-level MoE routing (Gale et al., 2022; Shazeer et al., 2017), Maglev Indexers operates on feature fields under explicit output and storage budgets. DMT (Luo et al., 2024) also assigns feature partitions to scale-up domains, but clusters fields offline and fuses independent towers late; Maglev learns the partition under per-stage budgets and conditions each stage on its predecessors, which is what enables depth scaling (Table 3). Meta Lattice (Luo et al., 2025) and Kunlun (Hou et al., 2026) are composite systems rather than atomic interaction backbones; Maglev can transform their constituent blocks but does not subsume their orthogonal components. See further discussion in § A.13.

Scope and limitations. Industrial results use proprietary data and one seed per configuration, with repeatability measured separately (§A.10.1); Maglev Rail's zero-bubble property holds under the standard equal-latency model and its measured throughput reflects real latency variation.

6 Conclusion

Maglev combines learned feature partitioning, cross-stage residual context, and hardware-local pipeline execution. Across industrial scaling experiments, public benchmarks, and three production launches, it improves recommendation quality and scaling efficiency without modifying backbone internals.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology, formulate and check mathematical claims, assist in writing proof sketches, and critique the interpretation and scope of reported results. We have not used generative AI tools for other tasks with required disclosure, and the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for proofreading, manuscript restructuring, tabular presentation, literature summarization, and adversarial review. We reviewed all AI-assisted work and manually verified generated proof sketches, calculations, citations, and suggested edits. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide pseudo code for Maglev Indexers and Maglev Induct implementations in § A.1 and § A.8. The public evaluation uses the BARS benchmark data directly; §A.6.1 reports the split, shared training settings, each configuration, and FLOPs.

Consistent with prior work (Zhang et al., 2024a; 2023; Zeng et al., 2024; Zhu et al., 2025b; Xu et al., 2025b; Shin et al., 2023; Yan et al., 2026; Zhai et al., 2024; Hou et al., 2026), our scaling-law study uses an internal dataset to avoid saturation artifacts of public benchmarks. The dataset cannot be open-sourced because of privacy constraints and Petabyte-scale storage requirements. To make the method auditable and reusable, we use public-baseline implementations in FuxiCTR (Zhu et al., 2020) from GitHub (rec), detailed hyperparameters for the industrial configurations (Appendix A.6.2), and a concrete Maglev reference implementation (Appendix A.8).

REFERENCES

Display advertising challenge | kaggle. https://www.kaggle.com/c/criteo-display-ad-challenge/overview/evaluation. (Accessed on 05/06/2024).

Wsdm - kkbox's music recommendation challenge | kaggle. URL https://www.kaggle.com/c/kkbox-music-recommendation-challenge. [Online; accessed 2026-08-05].

Nvidia gb200 nvl72. URL https://www.nvidia.com/en-us/data-center/gb200-nvl72/.

reczoo/fuxictr: A configurable, tunable, and reproducible library for ctr prediction https://fuxictr.github.io. URL https://github.com/reczoo/FuxiCTR. [Online; accessed 2026-05-01].

Newsha Ardalani, Carole-Jean Wu, Zeliang Chen, Bhargav Bhushanam, and Adnan Aziz. Understanding scaling laws for recommendation models. *arXiv preprint arXiv:2208.08489*, 2022.

Sangmin Bae, Bilge Acun, Chien-Yu Lin, Haroun Habeeb, Seungyeon Kim, Liang Luo, Junjie Wang, and Carole-Jean Wu. Hybrid architectures for language models: Systematic analysis and design insights, 2026. URL https://arxiv.org/abs/2510.04800.

Paul S Bradley, Kristin P Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, 20(0):0, 2000.

Renqin Cai, Dawei Sun, Yuanjun Yao, Zhiyong Wang, Velvin Fu, Maggie Zhuang, Yu Shi, Zhongnan Fang, Xuan Cao, Jing Qian, and Rui Li. Whale: A scalable unified model for recommendation with wukong-hstu architecture, 2026. URL https://arxiv.org/abs/2607.17017.

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

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, Han Zhang, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. Conditional memory via scalable lookup: A new axis of sparsity for large language models, 2026. URL https://arxiv.org/abs/2601.07372.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2022. URL https://arxiv.org/abs/2009.14794.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Souvik Debnath, Niloy Ganguly, and Pabitra Mitra. Feature weighting in content based recommendation system using social network analysis. In *Proceedings of the 17th international conference on World Wide Web*, pp. 1041–1042, 2008.

DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Junguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei

Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025. URL https://arxiv.org/abs/2512.02556.

Chenhao Feng, Haoli Zhang, Shakhzod Ali-Zade, Yanli Zhao, Liang Luo, Jennifer Cao, Lisen Deng, Siqiao Chen, Chenyu Zhao, Tristan Rice, et al. Freescale: Distributed training for sequence recommendation models with minimal scaling cost. *arXiv preprint arXiv:2604.24073*, 2026.

Zichuan Fu, Wentao Song, Yejing Wang, Xian Wu, Yefeng Zheng, Yingying Zhang, Derong Xu, Xuetao Wei, Tong Xu, and Xiangyu Zhao. Sliding window attention training for efficient large language models, 2025. URL https://arxiv.org/abs/2502.18845.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts, 2022. URL https://arxiv.org/abs/2211.15841.

Xin Gao, Sibasish Acharya, Sihui Han, Yongxiong Ren, Yanli Zhao, Liang Luo, Chucheng Wang, Pradeep Fernando, Saurabh Mishra, Siqi Yan, et al. Deck: Experiences on delta checkpointing for industrial recommendation systems. *Proceedings of the VLDB Endowment*, 18(12):4978–4990, 2025.

Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. Ai and memory wall, 2024. URL https://arxiv.org/abs/2403.14123.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL https://arxiv.org/abs/2312.00752.

Wei Guo, Hao Wang, Luankang Zhang, Jin Yao Chin, Zhongzhou Liu, Kai Cheng, Qiushi Pan, Yi Quan Lee, Wanqi Xue, Tingjia Shen, Kenan Song, Kefan Wang, Wenjia Xie, Yuyang Ye, Huifeng Guo, Yong Liu, Defu Lian, Ruiming Tang, and Enhong Chen. Scaling new frontiers: Insights into large recommendation models, 2024. URL https://arxiv.org/abs/2412.00714.

Xingzhuo Guo, Junwei Pan, Ximei Wang, Baixu Chen, Jie Jiang, and Mingsheng Long. On the embedding collapse when scaling up recommendation models. *arXiv preprint arXiv:2310.04400*, 2023.

Aaron Harlap, Deepak Narayanan, Amar Phanishayee, Vivek Seshadri, Nikhil Devanur, Greg Ganger, and Phil Gibbons. Pipedream: Fast and efficient pipeline parallel dnn training. *arXiv preprint arXiv:1806.03377*, 2018.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Bojian Hou, Xiaolong Liu, Xiaoyi Liu, Jiaqi Xu, Yasmine Badr, Mengyue Hang, Sudhanshu Chanpuriya, Junqing Zhou, Yuhang Yang, Han Xu, Qiuling Suo, Laming Chen, Yuxi Hu, Jiasheng Zhang, Huaqing Xiong, Yuzhen Huang, Chao Chen, Yue Dong, Yi Yang, Shuo Chang, Xiaorui Gan, Wenlin Chen, Santanu Kolay, Darren Liu, Jade Nie, Chunzhi Yang, Ellie Wen, Jiyan Yang, and Huayu Li. Kunlun: Establishing scaling laws for massive-scale recommendation systems through unified architecture design, 2026. URL https://arxiv.org/abs/2602.10016.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018. URL https://arxiv.org/abs/1608.06993.

Dmytro Ivchenko, Dennis Van Der Staay, Colin Taylor, Xing Liu, Will Feng, Rahul Kindi, Anirudh Sudarshan, and Shahin Sefati. Torchrec: a pytorch domain library for recommendation systems. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pp. 482–483, 2022.

Himanshu Jain, Yashoteja Prabhu, and Manik Varma. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 935–944, 2016.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kuaishou. Kuaishou user interest modeling challenge. https://www.kuaishou.com/activity/uimc. (Accessed on 08/07/2024).

Honghao Li, Yiwen Zhang, Yi Zhang, Hanwei Li, Lei Sang, and Jieming Zhu. Fcn: Fusing exponential and linear cross network for click-through rate prediction, 2025. URL https://arxiv.org/abs/2407.13349.

Pan Li and Alexander Tuzhilin. Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 331–339, 2020.

Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pp. 539–548. ACM, November 2019. doi: 10.1145/3357384.3357951. URL http://dx.doi.org/10.1145/3357384.3357951.

Mingfu Liang, Xi Liu, Rong Jin, Boyang Liu, Qiuling Suo, Qinghai Zhou, Song Zhou, Laming Chen, Hua Zheng, Zhiyuan Li, Shali Jiang, Jiyan Yang, Xiaozhen Xia, Fan Yang, Yasmine Badr, Ellie Wen, Shuyu Xu, Hansey Chen, Zhengyu Zhang, Jade Nie, Chunzhi Yang, Zhichen Zeng, Weilin Zhang, Xingliang Huang, Qianru Li, Shiquan Wang, Evelyn Lyu, Wenjing Lu, Rui Zhang, Wenjun Wang, Jason Rudy, Mengyue Hang, Kai Wang, Yinbin Ma, Shuaiwen Wang, Sihan Zeng, Tongyi Tang, Xiaohan Wei, Longhao Jin, Jamey Zhang, Marcus Chen, Jiayi Xu, Angie Huang, Xihuan Zeng, Chi Zhang, Zhengli Zhao, Jared Yang, Qiang Jin, Xian Chen, Amit Anand Amlesahwaram, Lexi Song, Liang Luo, Yuchen Hao, Nan Xiao, Yavuz Yetim, Luoshang Pan, Gaoxiang Liu, Yuxi Hu, Yuzhen Huang, Jackie Xu, Rich Zhu, Xin Zhang, Yiqun Liu, Hang Yin, Yuxin Chen, Buyun Zhang, Xiaoyi Liu, Xingyuan Wang, Wenguang Mao, Zhijing Li, Zhehui Zhou, Feifan Gu, Qin Huang, Chonglin Sun, Nancy Yu, Shuo Gu, Shupin Mao, Benjamin Au, Jingzheng Qin, Peggy Yao, Jae-Woo Choi, Bin Gao, Ernest Wang, Lei Zhang, Wen-Yen Chen, Ted Lee, Yujie Zha, Yi Meng,

Alex Gong, Edison Gao, Jack Hsueh, Jie Zheng, Alireza Vahdatpour, Yiping Han, Yantao Yao, Toshinari Kureha, Shuo Chang, Musharaf Sultan, John Bocharov, Sagar Chordia, Xiaorui Gan, Peng Sun, Rocky Liu, Bo Long, Wenlin Chen, Santanu Kolay, and Huayu Li. External large foundation model: How to efficiently serve trillions of parameters for online ads recommendation, 2025. URL https://arxiv.org/abs/2502.17494.

Bin Liu, Niannan Xue, Huifeng Guo, Ruiming Tang, Stefanos Zafeiriou, Xiuqiang He, and Zhenguo Li. Autogroup: Automatic feature grouping for modelling explicit high-order feature interactions in ctr prediction. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pp. 199–208, 2020a.

Bin Liu, Chenxu Zhu, Guilin Li, Weinan Zhang, Jincai Lai, Ruiming Tang, Xiuqiang He, Zhenguo Li, and Yong Yu. Autofis: Automatic feature interaction selection in factorization models for click-through rate prediction. In *proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2636–2645, 2020b.

Zikun Liu, Liang Luo, Qianru Li, Zhengyu Zhang, Wei Ling, Jingyi Shen, Zeliang Chen, Yaning Huang, Jingxian Huang, Abdallah Aboelela, Chonglin Sun, Feifan Gu, Fenggang Wu, Hang Qu, Huayu Li, Jill Pan, Kaidi Pei, Laming Chen, Longhao Jin, Qin Huang, Tongyi Tang, Varna Puvvada, Wenlin Chen, Xiaohan Wei, Xu Cao, Yantao Yao, Yuan Jin, Yunchen Pu, Yuxin Chen, Zijian Shen, Zhengkai Zhang, Dong Liang, and Ellie Wen. Solaris: Speculative offloading of latent-based representation for inference scaling, 2026. URL https://arxiv.org/abs/2604.12110.

Liang Luo, Ming Liu, Jacob Nelson, Luis Ceze, Amar Phanishayee, and Arvind Krishnamurthy. Motivating in-network aggregation for distributed deep neural network training. In *Workshop on Approximate Computing Across the Stack*, 2017.

Liang Luo, Jacob Nelson, Luis Ceze, Amar Phanishayee, and Arvind Krishnamurthy. 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.

Liang Luo, Jacob Nelson, Luis Ceze, Amar Phanishayee, and Arvind Krishnamurthy. Parameter box: High performance parameter servers for efficient distributed deep neural network training, 2020. URL https://arxiv.org/abs/1801.09805.

Liang Luo, Buyun Zhang, Michael Tsang, Yinbin Ma, Ching-Hsiang Chu, Yuxin Chen, Shen Li, Yuchen Hao, Yanli Zhao, Guna Lakshminarayanan, et al. Disaggregated multi-tower: Topology-aware modeling technique for efficient large-scale recommendation. *arXiv preprint arXiv:2403.00877*, 2024.

Liang Luo, Yuxin Chen, Zhengyu Zhang, Mengyue Hang, Andrew Gu, Buyun Zhang, Boyang Liu, Chen Chen, Chengze Fan, Dong Liang, Fan Yang, Feifan Gu, Huayu Li, Jade Nie, Jiayi Xu, Jiyan Yang, Jongsoo Park, Laming Chen, Longhao Jin, Qianru Li, Qin Huang, Shali Jiang, Shiwen Shen, Shuaiwen Wang, Sihan Zeng, Siyang Yuan, Tongyi Tang, Weilin Zhang, Wenjun Wang, Xi Liu, Xiaohan Wei, Xiaozhen Xia, Yuchen Hao, Yunlong He, Yasmine Badr, Zeliang Chen, Maxim Naumov, Yantao Yao, Wenlin Chen, Santanu Kolay, GP Musumeci, and Ellie Dingqiao Wen. Meta lattice: Model space redesign for cost-effective industry-scale ads recommendations, 2025. URL https://arxiv.org/abs/2512.09200.

Liang Luo, Yinbin Ma, Quanyu Zhu, Vasiliy Kuznetsov, Yuxin Chen, Neng Shi, Jian Jiao, Jiecao Yu, Buyun Zhang, Tongyi Tang, Xiaohan Wei, Yanli Zhao, Zeliang Chen, Yuchen Hao, Venkatesh Ranganathan, Sandeep Parab, Yantao Yao, Maxim Naumov, Chunzhi Yang, Shen Li, Ellie Wen, Wenlin Chen, Santanu Kolay, and Chunqiang Tang. Loka: Low-precision kernel applications for recommendation models at scale, 2026. URL https://arxiv.org/abs/2605.10886.

Dongyang Ma, Yan Wang, and Lan Tian. Block-attention for efficient prefilling. *arXiv preprint arXiv:2409.15355*, 2024.

Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. Finalmlp: An enhanced two-stream mlp model for ctr prediction. *arXiv preprint arXiv:2304.00902*, 2023.

Dheevatsa Mudigere, Yuchen Hao, Jianyu Huang, Andrew Tulloch, Srinivas Sridharan, Xing Liu, Mustafa Ozdal, Jade Nie, Jongsoo Park, Liang Luo, et al. High-performance, distributed training of large-scale deep learning recommendation models. *arXiv preprint arXiv:2104.05158*, 2021.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm, 2021. URL https://arxiv.org/abs/2104.04473.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Jackson Petty, Sjoerd van Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. The impact of depth on compositional generalization in transformer language models. In *North American Chapter of the Association for Computational Linguistics*, 2023. URL https://api.semanticscholar.org/CorpusID:264820247.

Penghui Qi, Xinyi Wan, Guangxing Huang, and Min Lin. Zero bubble pipeline parallelism, 2023. URL https://arxiv.org/abs/2401.10241.

Penghui Qi, Xinyi Wan, Nyamdavaa Amar, and Min Lin. Pipeline parallelism with controllable memory. *Advances in Neural Information Processing Systems*, 37:46539–46566, 2024.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

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

Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 4104–4113, 2021.

Kyuyong Shin, Hanock Kwak, Su Young Kim, Max Nihlén Ramström, Jisu Jeong, Jung-Woo Ha, and Kyung-Min Kim. 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.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Min Si, Pavan Balaji, Yongzhou Chen, Ching-Hsiang Chu, Adi Gangidi, Saif Hasan, Subodh Iyengar, Dan Johnson, Bingzhe Liu, Regina Ren, Deep Shah, Ashmitha Jeevaraj Shetty, Greg Steinbrecher, Yulun Wang, Bruce Wu, Xinfeng Xie, Jingyi Yang, Mingran Yang, Kenny Yu, Minlan Yu, Cen Zhao, Wes Bland, Denis Boyda, Suman Gumudavelli, Prashanth Kannan, Cristian Lumezanu, Rui Miao, Zhe Qu, Venkat Ramesh, Maxim Samoylov, Jan Seidel, Srikanth Sundaresan, Feng Tian, Qiye Tan, Shuqiang Zhang, Yimeng Zhao, Shengbao Zheng, Art Zhu, and Hongyi Zeng. Collective communication for 100k+ gpus, 2026. URL https://arxiv.org/abs/2510.20171.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 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.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers, 2020. URL https://arxiv.org/abs/2011.04006.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture, 2025. URL https://arxiv.org/abs/2510.26692.

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

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 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.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. URL https://arxiv.org/abs/2006.04768.

Yichao Wang, Huifeng Guo, Bo Chen, Weiwen Liu, Zhirong Liu, Qi Zhang, Zhicheng He, Hongkun Zheng, Weiwei Yao, Muyu Zhang, et al. Causalint: Causal inspired intervention for multi-scenario recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4090–4099, 2022.

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

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Zhenda Xie, Yixuan Wei, Huanqi Cao, Chenggang Zhao, Chengqi Deng, Jiashi Li, Damai Dai, Huazuo Gao, Jiang Chang, Kuai Yu, Liang Zhao, Shangyan Zhou, Zhean Xu, Zhengyan Zhang, Wangding Zeng, Shengding Hu, Yuqing Wang, Jingyang Yuan, Lean Wang, and Wenfeng Liang. mhc: Manifold-constrained hyper-connections, 2026. URL https://arxiv.org/abs/2512.24880.

Songpei Xu, Shijia Wang, Da Guo, Xianwen Guo, Qiang Xiao, Bin Huang, Guanlin Wu, and Chuanjiang Luo. Climber: Toward efficient scaling laws for large recommendation models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, CIKM '25, pp. 6193–6200, New York, NY, USA, 2025a. Association for Computing Machinery. ISBN 9798400720406. doi: 10.1145/3746252.3761561. URL https://doi.org/10.1145/3746252.3761561.

Songpei Xu, Shijia Wang, Da Guo, Xianwen Guo, Qiang Xiao, Bin Huang, Guanlin Wu, and Chuanjiang Luo. Climber: Toward efficient scaling laws for large recommendation models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6193–6200, 2025b.

Yichen Xu, Yanqiao Zhu, Feng Yu, Qiang Liu, and Shu Wu. Disentangled self-attentive neural networks for click-through rate prediction. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, CIKM '21, pp. 3553–3557. ACM, October 2021. doi: 10.1145/3459637.3482088. URL http://dx.doi.org/10.1145/3459637.3482088.

Bencheng Yan, Pengjie Wang, Kai Zhang, Feng Li, Hongbo Deng, Jian Xu, and Bo Zheng. Apg: Adaptive parameter generation network for click-through rate prediction. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24740–24752. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9cd0c57170f48520749d5ae62838241f-Paper-Conference.pdf.

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. Unlocking scaling law in industrial recommendation systems with a three-step paradigm based large user model. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*, WSDM '26, pp. 798–807, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400722929. doi: 10.1145/3773966.3777970. URL https://doi.org/10.1145/3773966.3777970.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2025. URL https://arxiv.org/abs/2412.06464.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition. *arXiv preprint arXiv:2402.15220*, 2024.

Da Yu, Edith Cohen, Badih Ghazi, Yangsibo Huang, Pritish Kamath, Ravi Kumar, Daogao Liu, and Chiyuan Zhang. Scaling embedding layers in language models. *arXiv preprint arXiv:2502.01637*, 2025.

Zhichen Zeng, Xiaolong Liu, Mengyue Hang, Xiaoyi Liu, Qinghai Zhou, Chaofei Yang, Yiqun Liu, Yichen Ruan, Laming Chen, Yuxin Chen, Yujia Hao, Jiaqi Xu, Jade Nie, Xi Liu, Buyun Zhang, Wei Wen, Siyang Yuan, Kai Wang, Wen-Yen Chen, Yiping Han, Huayu Li, Chunzhi Yang, Bo Long, Philip S. Yu, Hanghang Tong, and Jiyan Yang. Interformer: Towards effective heterogeneous interaction learning for click-through rate prediction, 2024. URL https://arxiv.org/abs/2411.09852.

Daochen Zha, Louis Feng, Liang Luo, Bhargav Bhushanam, Zirui Liu, Yusuo Hu, Jade Nie, Yuzhen Huang, Yuandong Tian, Arun Kejariwal, et al. Pre-train and search: Efficient embedding table sharding with pre-trained neural cost models. *Proceedings of Machine Learning and Systems*, 5, 2023.

Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152*, 2024.

Buyun Zhang, Liang Luo, Xi Liu, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, et al. Dhen: A deep and hierarchical ensemble network for large-scale click-through rate prediction. *arXiv preprint arXiv:2203.11014*, 2022.

Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. Wukong: Towards a scaling law for large-scale recommendation. *arXiv preprint arXiv:2403.02545*, 2024a.

Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. Scaling law of large sequential recommendation models, 2023. URL https://arxiv.org/abs/2311.11351.

Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. Scaling law of large sequential recommendation models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, RecSys '24, pp. 444–453, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400705052. doi: 10.1145/3640457.3688129. URL https://doi.org/10.1145/3640457.3688129.

Xin Zhang, Quanyu Zhu, Liangbei Xu, Zain Huda, Wang Zhou, Jin Fang, Dennis van der Staay, Yuxi Hu, Jade Nie, Jiyan Yang, and Chunzhi Yang. Two-dimensional sparse parallelism for large scale deep learning recommendation model training, 2025. URL https://arxiv.org/abs/2508.03854.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections, 2025a. URL https://arxiv.org/abs/2409.19606.

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. Rankmixer: Scaling up ranking models in industrial recommenders, 2025b. URL https://arxiv.org/abs/2507.15551.

Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. Fuxictr: An open benchmark for click-through rate prediction. *arXiv preprint arXiv:2009.05794*, 2020.

Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. BARS: towards open benchmarking for recommender systems. In Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (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, 2022. doi: 10.1145/3477495.3531723. URL https://doi.org/10.1145/3477495.3531723.

Jinfeng Zhuang, Yinrui Li, Runze Su, Ke Xu, Zhixuan Shao, Kungang Li, Ling Leng, Han Sun, Meng Qi, Yixiong Meng, et al. On the practice of deep hierarchical ensemble network for ad conversion rate prediction. In *Companion Proceedings of the ACM on Web Conference 2025*, pp. 671–680, 2025.

A Appendix

A.1 Hard Maglev Indexers Partitioning Algorithm

Algorithm 1 Soft-to-Hard Maglev Indexers Partitioning

Require: Affinity matrix $\mathcal{A}^T \in \mathbb{R}^{(\sum_i P_i)\times F}$, output widths $[P_1, \ldots, P_K]$, and storage budgets $[S_1, \ldots, S_K]$
Ensure: Partitions $\{\mathcal{F}_i\}_{i=1}^K$ satisfying $|\mathcal{F}_i| = S_i$ and $\bigcup_i \mathcal{F}_i = \{0, \ldots, F-1\}$
  Validate $1 \le S_i \le F$ and $\sum_i S_i \ge F$
  Split $\mathcal{A}^T$ into $\{\mathcal{A}_i^T\}_{i=1}^K$, where $\mathcal{A}_i^T \in \mathbb{R}^{P_i \times F}$
  for $i = 1$ to K **do**
    $v_i \leftarrow \mathrm{mean}(|\mathcal{A}_i^T|, \mathrm{axis} = 0)$
    $\mathcal{F}_i \leftarrow \mathrm{TopK}_{S_i}(v_i).\mathrm{Indices}$
  end for
  $c_f \leftarrow \sum_i \mathbf{1}[f \in \mathcal{F}_i]$ for every feature f
  $\mathcal{M} \leftarrow \{f : c_f = 0\}$
  for each $f \in \mathcal{M}$ **do**
    Choose the lowest-affinity selected occurrence (i, g) such that $c_g > 1$
    $\mathcal{F}_i \leftarrow (\mathcal{F}_i \setminus \{g\}) \cup \{f\}$
    $c_g \leftarrow c_g - 1; \quad c_f \leftarrow 1$
  end for
  return $\{\mathcal{F}_i\}_{i=1}^K$

Algorithm 1 preserves every stage's storage budget while guaranteeing global feature coverage. To see that a replacement is always available, let $T = \sum_i S_i$ be the number of assignment slots and let U be the number of distinct assigned features. There are $F - U$ missing features and $T - U$ redundant assignments. Since $T \ge F$, we have $T - U \ge F - U$. Each replacement removes one redundant assignment and increases U by one, so the procedure terminates after exactly $F - U$ replacements. Importantly, the coverage condition applies to the storage budgets S_i; it imposes no requirement that $\sum_i P_i \ge F$.

When $\sum_i S_i > F$, the remaining assignments intentionally reuse fields across stages. Such reuse duplicates only the raw input; each assignment denotes an independently parameterized stage-local embedding table rather than a synchronized parameter replica.

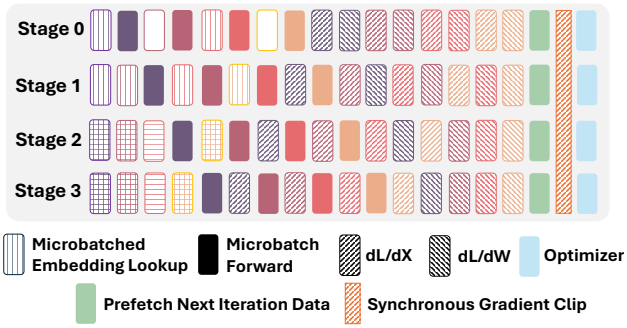

Figure 5: Maglev Rail's Zero Bubble schedule under the standard equal-operation-latency pipeline model.

A.2 Maglev Rail Optimality Proof

Consider a P-stage pipeline ($P = K$ for a K-stage Maglev model) with M microbatches. Under the equal-operation-latency model, Maglev Rail eliminates structural bubbles in the scheduled embedding, forward, and backward region. After all current-iteration weight gradients have been drained, the stages compute the exact global gradient norm, apply clipping, and perform the optimizer update before the next iteration begins. This common post-backward tail is excluded from the theoretical bubble rate but included in measured end-to-end throughput. Thus, Maglev Rail remains compatible with exact synchronous gradient clipping without speculative execution or rollback.

Relation to the ZB schedules. ZB2P (Qi et al., 2023) reaches zero bubble with a parallelogram schedule whose cool-down overlaps the next iteration's warm-up. An exact synchronous-clipping barrier forbids that overlap and turns both the warm-up and the cool-down into idle time, so filling ZB2P's warm-up with lookups still leaves its drain. ZB-H1 finishes all stages together and therefore meets the barrier without a drain, but leaves $(P-1)(T_F + T_B - T_W)$ idle time per stage in the warm-up. At $M = P$, Maglev Rail keeps ZB-H1's order of forward, IGrad and WGrad operations and places each stage's P embedding lookups in exactly those slots. This requires every stage to own lookups that depend on no other stage, which the Maglev Indexers partition provides; with lookups fused into the forward, as in a monolithic recommender, the same order retains ZB-H1's bubble (Tables 4 and 5).

Table 4: Pipeline schedules under an exact synchronous-clipping barrier between iterations (no optimizer post-validation), in the equal-latency model. All columns are relative to 1F1B.

Pipeline Schedule	Bubble (rel.)	Peak Live Activations	Comms. Volume
1F1B	1	1	1
ZB-H1 / ZB1P	$1/3$	1	1
ZB2P	$1/3$	2	1
ZBV-2	$1/6$	1	2
V-Min	$2/9$	$\mathbf{1/3}$	2
V-Half	$1/6$	$1/2$	2
Maglev Rail	**0**	1	**1**

Bubble rates are theoretical rates under the standard equal-latency pipeline model (Harlap et al., 2018; Qi et al., 2023; 2024; Shoeybi et al., 2020; Narayanan et al., 2021). Baseline rates use the conventional three-operation model (F, B, W); with each stage's embedding lookup fused into F, the ZB-H1/ZB1P and ZB2P rows rise to $1/2$. The Maglev Rail row is derived below.

Activation-state liveness. Let $M = P$, where M is the number of microbatches and P is the pipeline depth. An eagerly scheduled embedding lookup materializes the ordinary stage-input activation for its microbatch; it does not create a second copy. Its storage is subsequently accounted for as part of that microbatch's forward activation record and is released after the corresponding backward operation. Consequently, each stage retains at most one activation record per in-flight microbatch and at most P

records overall. This matches the peak live-activation-state bound of 1F1B configured with the same $M = P$. The comparison concerns activation records rather than persistent parameters, embedding tables, or optimizer state, which are common to both schedules.

Pipeline-boundary communication. For each microbatch, every pipeline boundary carries one forward activation tensor and one backward input-gradient tensor. Thus, for the same partition and $M = P$, Maglev Rail matches 1F1B's pipeline-boundary communication volume. This comparison excludes within-stage embedding collectives, data-parallel reductions, gradient-norm reduction, and optimizer traffic.

Full-Trajectory Zero Bubble with Synchronous Gradient Clipping: We provide a proof sketch for the scheduled embedding, forward, and backward trajectory under the standard equal-operation-latency model, where embedding lookup, forward, IGrad, and WGrad each occupy one scheduled slot.

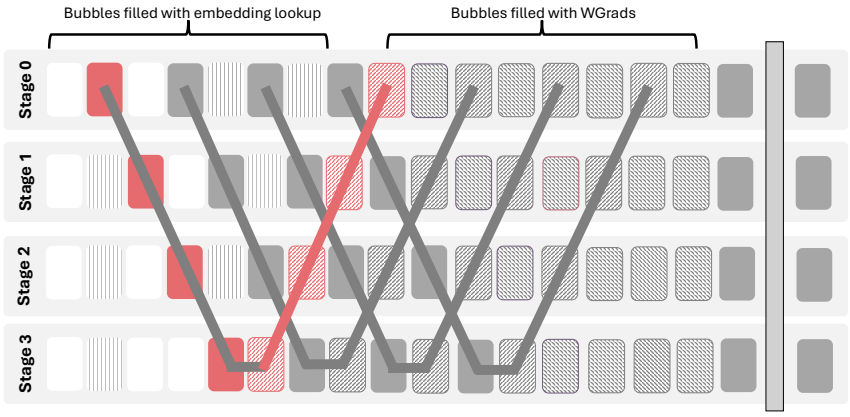

Figure 6: Proof sketch for Maglev Rail's zero bubble claim.

Consider the execution timeline before the first backward signal is propagated (denoted by the "red line" in Figure 6). Maglev Rail's scheduling rules naturally produce a "V-shape" execution block for each microbatch. These V-shapes are geometrically offset by two timesteps and strictly avoid conflicts. Prior to the red line, the number of idle execution slots (bubbles) exactly equals the number of required embedding lookups for the current iteration's microbatches. Because every forward pass depends only on its corresponding embedding lookup (which has no inter-microbatch dependency)—and there are at least as many available bubble slots as there are forward slots—we can schedule an embedding lookup eagerly into each available bubble. Thus, the entire warmup schedule is provably bubble-free up to the red line. Crucially, this construction relies only on the current iteration's embedding lookups; it does not require WGrads from the previous iteration, which synchronous gradient clipping has already drained at the inter-iteration barrier.

The $M = P$ construction is the smallest zero-bubble rectangle, not a restriction on microbatch count. For any $M = kP$, $k \geq 1$, we concatenate k copies of this rectangle before the single iteration-level clipping barrier. Because every stage boundary is aligned and every slot is occupied, adjacent rectangles join into one seamless $4M$-slot rectangle without introducing bubbles. Each stage still holds at most P activation records, each $k\times$ smaller at fixed global batch size, consistent with the $M{=}4P$ column of Table 5.

Backward-region availability. For a fixed stage, order microbatches by the arrival of their output gradients. The IGrad and WGrad for a microbatch share the same readiness prerequisites: its saved forward activation and incoming output gradient. Maglev Rail prioritizes the dependency-critical IGrad, leaving its paired WGrad in the ready queue. In the V-shaped construction, consecutive IGrad operations expose at most one complementary local slot. After the first IGrad, one WGrad is ready. Inductively, each subsequent IGrad adds one ready WGrad, while each complementary slot consumes at most one. Hence the ready queue is nonempty whenever a complementary slot occurs, and every such slot is filled by a WGrad, including the terminal WGrad after the final IGrad.

Table 5: Simulated bubble rate (%, idle stage-time before the clipping barrier) at $P = 16$. Each column changes one parameter of the equal-latency model ($T_E = T_F = T_B = T_W$, $M = P$, no transfer delay): microbatch count, lookup latency, WGrad latency, lognormal per-operation jitter (coefficient of variation; mean of 100 paired draws), and pipeline-boundary delay C, except Measured, which uses latencies from a production run (T_E=1.52, T_B=1.03, T_W=0.97, in units of T_F). Each schedule's operation order is computed from nominal latencies and then executed. Records: peak live activation records per stage.

Schedule	Records	M=P	M=4P	T_E=$\frac{1}{4}T_F$	T_E=2T_F	T_W=$\frac{1}{2}T_F$	CV 10%	CV 20%	C=0.1T_F	Measured
1F1B (lookups fused into F)	16	48.4	19.0	48.4	48.4	48.4	49.6	50.9	50.7	48.4
ZB-H1 (lookups fused into F)	16	31.9	10.5	31.6	36.0	40.1	34.2	36.7	35.9	34.9
Maglev Rail	16	0.0	0.0	17.8	0.0	11.8	6.9	13.2	8.3	1.3

Together with the warmup construction, each stage therefore executes exactly P embedding, P forward, P IGrad, and P WGrad operations without an idle slot. Since the offset V-shaped trajectories do not conflict, all stages complete their scheduled work at timestamp $4P$.

The zero-bubble result covers the scheduled embedding, forward, and backward trajectory through arrival at the synchronous clipping barrier. At timestamp $4P$, every stage has completed its current-iteration weight gradients and can enter the exact global gradient-norm reduction simultaneously; hence clipping introduces no drain or waiting bubble. The norm reduction, clipping arithmetic, and optimizer update have nonzero latency, but constitute synchronized useful work rather than pipeline idle time and are included in measured end-to-end throughput.

Sensitivity to unequal latencies. Table 5 relaxes the equal-latency assumption in simulation. Maglev Rail stays bubble-free whenever each lookup takes at least as long as a forward pass and IGrad and WGrad take equal time; longer lookups add no bubble (0.0% up to $T_E = 4T_F$). With lighter lookups or unequal IGrad and WGrad, it leaves approximately the difference idle. At latencies measured in a production run (lookup, forward, IGrad, and WGrad about 15, 9.9, 10.2, and 9.6 ms per microbatch), the simulated bubble is 0.8–1.3% for $P \in \{16, 32\}$ and $M \in \{P, 2P, 4P\}$, against 11.6–35.6% for ZB-H1. Its bubble rate grows gradually with per-operation jitter and boundary-transfer delay; on the production model, stage-to-stage transfers take about $0.02\,T_F$, well below the $C = 0.1\,T_F$ in Table 5. In every setting it is at most 57% of ZB-H1's, with the same peak of P activation records. At $P = 32$, all entries change by at most 1.2 points. The lookup, which grows with the sparse feature count, exceeds $T_F + T_B - T_W \approx 10.5$ ms, so the zero-bubble condition holds at every P; the lookup could shrink by about 30% before it fails.

A.3 Maglev Rail Optimizations

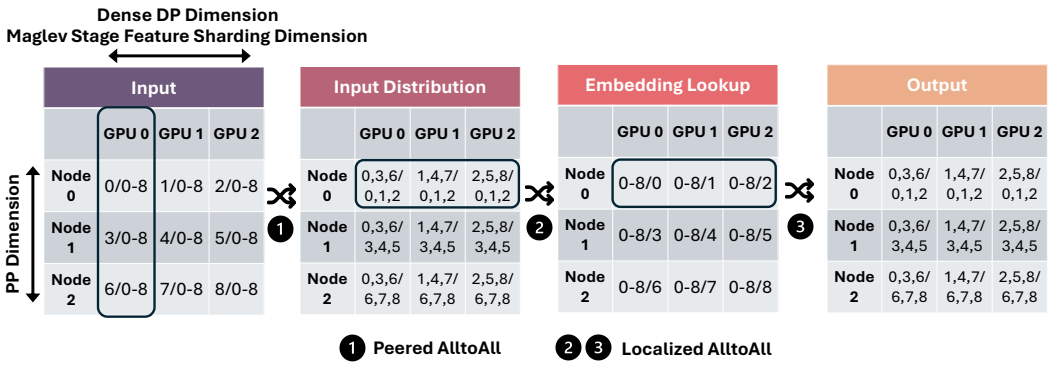

Figure 7: The Pipeline Parallel Table-wise Column-wise Hierarchical Sharding Scheme for Maglev.

A.3.1 PPTwCw Sharding

Each iteration begins with raw IDs and feature values supplied by the central input service. The Maglev Indexers assignment determines the destination stage or stages for each field, and raw inputs

assigned more than once are fanned out accordingly. Each destination performs lookup using its own independently learned embedding-table instance, sharded only within that stage's HSD. Thus, no individual parameter instance is redundantly stored across HSDs, and repeated assignments require no cross-stage parameter or gradient synchronization.

All stage-local table instances are included in the reported sparse-parameter and storage budgets, while raw-input routing is included in measured end-to-end throughput.

PPTwCw routes the minibatch inputs to their stage-local embedding owners using parallel peered AlltoAll operations between corresponding local ranks, followed by parallel intra-HSD AlltoAll operations. After routing, each pipeline holds its minibatch and each stage holds only the inputs assigned to it. This keeps embedding lookup local and avoids a global exchange of embedding activations.

We provide a simplified example of PPTwCw in Figure 7, where a total of 9 features are sharded evenly across 3 HSDs, each with 3 GPUs. We use the notation `minibatch id/feature id` to denote the current GPU holds the data for `minibatch id` and feature shard `feature id`.

A.3.2 Asynchronous Communication Awaitable

Algorithm 2 Asynchronous Communication Awaitable

output_tensor $\leftarrow$ torch.empty($\ldots$, device = "cuda")
handle $\leftarrow$ dist.all_to_all_single(output_tensor, $\ldots$, async_op = True)
data $\leftarrow$ AsyncCommAwaitable(
    dist_async_handle = handle,
    tensor = output_tensor,
    callback = $\lambda\, t : t$.add_(bias)
)
def f(data):
    $\ldots$ {Independent computation can be inserted here}
    $y \leftarrow$ F.gelu(data) {Comm sync and callback are triggered here}
return y
$y \leftarrow f$(data)

We introduce a novel infrastructure primitive, Asynchronous Communication Awaitables (ACAs), to gracefully resolve the trade-off between explicitly blocking on communication (which halts computation) and passing communication handles deeply into downstream functions (which requires extensive, invasive code refactoring). As outlined in Algorithm 2, ACAs abstract the synchronization logic, allowing the framework to implicitly await communication only at the exact moment the tensor data is mathematically required, thereby maximizing compute-communication overlap automatically.

A.4 Maglev Rail Performance Sweep

Table 6 isolates Maglev Rail's schedule against ZB2P under exact synchronous gradient clipping. Both schedules use $M = P$, the same global batch size, model partition, clipping threshold, and optimizer configuration. The ZB2P baseline must drain all deferred current-iteration WGrads before entering the exact global-norm barrier; the resulting idle time is included. Reported end-to-end throughput includes norm reduction, clipping, and the optimizer step for both schedules. Under this barrier ZB2P and ZB-H1 have the same structural idle time (Table 4), so the comparison isolates the effect of filling the warm-up.

Across all 12 measured configurations, Maglev Rail improves end-to-end throughput over ZB2P by 6.3%–11.7%. These results establish consistency across the evaluated model sizes, GPU counts, and pipeline depths; the sweep is not designed to isolate monotonic effects of model size or topology.

Table 6: Isolated effect of Maglev Rail's zero-bubble schedule.

Model	**GPUs**	**PP** (Maglev Stages)	**DP** (8 GPUs/host)	**Improvement** (Synchronous Gradient Clipping)
13B	128	16	8	**7.8%**
13B	128	32	4	**7.9%**
13B	256	16	16	**6.3%**
13B	256	32	8	**6.3%**
24B	128	16	8	**8.4%**
24B	128	32	4	**9.2%**
24B	256	16	16	**7.6%**
24B	256	32	8	**9.1%**
64B	128	16	8	**9.0%**
64B	128	32	4	**6.8%**
64B	256	16	16	**11.7%**
64B	256	32	8	**10.9%**

A.5 Partition Stability

Figure 8 tracks 300 affinity snapshots over 250K training steps on time-ordered data. The affinity values stabilize after the initial warmup, and the resulting feature assignments remain coherent across the observed training trajectory. This experiment evaluates temporal stability for existing fields; it does not evaluate incremental placement of newly introduced fields.

Partition lifecycle. Maglev Indexers partitions feature fields rather than entities or categorical IDs, so embeddings update normally while the field-to-stage assignment remains fixed. New fields are handled through the standard model-refresh process and a new hours-scale proxy run. Only the induced partition is transferred; the target hard model contains no trainable affinity matrix.

Pretraining without a proxy. The shrunk proxy is a cost choice rather than a requirement. Because the affinities stabilize early, the pretraining run can instead be the full-size model with Soft Indexers, trained until $\mathcal{A}$ stabilizes and then resharded into the Maglev Rail form: each stage keeps its interaction modules, the embedding tables of its selected fields $\mathcal{F}_i$ move into its HSD (copied to every stage that selects the field), and its compression W_i is initialized from the rows of $\mathcal{A}_i$ indexed by $\mathcal{F}_i$. Training then continues in the Maglev Rail form. This removes any size mismatch between the run that learns the partition and the target model, at the cost of running the soft phase with the global embedding exchange that Maglev Rail otherwise avoids.

A.6 Hyperparameters

A.6.1 Public Dataset

Validation-selected public configurations

Backbone	Dataset	Native L	Maglev K	p
RankMixer	KKBox	3	3	13
RankMixer	Criteo	9	9	72
RankMixer	KuaiVideo	9	9	11
FinalMLP	KKBox	6	6	7
FinalMLP	Criteo	2	2	13
FinalMLP	KuaiVideo	3	3	4
AutoInt	KKBox	2	2	8
AutoInt	Criteo	2	2	34
AutoInt	KuaiVideo	3	3	3
DESTINE	KKBox	3	3	5
DESTINE	Criteo	3	3	13
DESTINE	KuaiVideo	3	3	3
FiGNN	KKBox	4	4	13
FiGNN	Criteo	2	2	39
FiGNN	KuaiVideo	2	2	7

Here, L is the native backbone depth, K is the number of Maglev stages, and p is the Indexer output width per stage and is distinct from the number of raw feature fields assigned to that stage. Configurations were selected using validation data under equal tuning budgets, then evaluated with 12 matched seeds. Criteo is split sequentially 72/18/10 into training, validation, and test; KKBox and KuaiVideo use the BARS splits. Both arms use batch size 4,096 on Criteo and KKBox and 8,192 on KuaiVideo. Forward FLOPs differ by at most 1.09% within a native/+Maglev pair.

A.6.2 JOINT SPARSE–DENSE GRID

Grid design. All 12 arms (10 architecture families and the two Maglev transformations) share the data, optimizer, batch size, and completion protocol, and are trained at the same four sparse capacities (104B to 410B, grown by widening every field's embedding 1–4×) over overlapping compute ranges; every arm covers 6–31 GFLOPs. Each family's grid starts from its reference configuration and scales it proportionally. We match budgets (G, P) rather than hyperparameters, because hyperparameters are not commensurable across architectures: the tables below list each family's own sizing knobs (e.g., DLRM is sized only by its top-MLP and compression widths), and a Maglev stage's Rank compresses $|X_{i-1}| + P_i + O$ tokens rather than all F features. MLP widths also scale with each configuration; the tables omit them for presentation clarity, but the reported FLOPs include them. Maglev receives no additional tuning budget, and every arm has 16 configurations, four per sparse capacity (15 for DLRM).

Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	RLL (%)
104	3.47	4	72	72	32	0.0000
104	6.20	8	72	72	32	0.1484
104	11.67	16	72	72	32	0.1074
104	31.67	16	128	128	64	0.2062
206	3.91	4	72	72	32	0.1041
206	12.10	16	72	72	32	0.1904
206	32.81	16	128	128	64	0.3054
206	43.41	16	128	128	128	0.3560
308	4.34	4	72	72	32	0.1759
308	12.52	16	72	72	32	0.4624
308	19.64	10	80	80	64	0.2780
308	33.95	16	128	128	64	0.3828
410	4.76	4	72	72	32	0.1968
410	12.95	16	72	72	32	0.3239
410	31.42	16	128	128	32	0.4036
410	47.40	16	128	128	128	0.5020

Maglev-Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	indexers	RLL (%)
104	3.81	4	16	64	64	48	0.0779
104	7.30	8	16	64	64	48	0.2241
104	15.38	8	32	128	96	96	0.3475
104	33.67	16	32	128	96	128	0.4843
206	3.94	4	16	64	64	48	0.1414
206	8.64	8	32	64	64	96	0.3487
206	15.76	8	32	128	96	96	0.4422
206	33.66	16	64	128	96	128	0.6020
308	4.06	4	16	64	64	48	0.2114
308	9.00	8	32	64	64	96	0.4121
308	16.13	8	32	128	96	96	0.5077
308	35.47	12	32	128	96	128	0.6562
410	4.45	8	0	48	32	72	0.3067
410	9.35	8	32	64	64	96	0.4532
410	16.50	8	32	128	96	96	0.5465
410	36.72	16	32	128	96	128	0.6624

DHEN

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	Attn Feats	RLL (%)
104	4.42	4	72	72	32	32	0.0504
104	8.02	8	72	72	32	32	0.1819
104	18.81	8	128	128	32	64	0.2672
104	36.95	16	128	128	32	64	0.3356
206	4.93	4	72	72	32	32	0.1570
206	19.54	8	128	128	32	64	0.4010
206	37.67	16	128	128	32	64	0.3021
206	44.13	16	128	128	64	64	0.4340
308	5.56	4	48	48	48	32	0.0961
308	9.02	8	72	72	32	32	0.3447
308	16.22	16	72	72	32	32	0.3213
308	45.20	16	128	128	64	64	0.4920
410	5.58	4	72	72	32	32	0.2900
410	11.31	8	72	72	64	32	0.2664
410	25.51	10	128	128	32	64	0.3184
410	46.27	16	128	128	64	64	0.4110

Maglev-DHEN

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	Attn Feats	indexers	RLL (%)
104	4.81	8	16	32	32	32	64	0.1295
104	10.90	8	96	96	48	32	96	0.2855
104	23.69	8	32	128	32	64	96	0.3598
104	34.01	16	64	128	64	64	64	0.4914
206	5.15	8	16	32	32	32	64	0.2333
206	7.23	8	32	64	32	32	64	0.3226
206	15.87	8	32	96	32	64	64	0.4165
206	34.69	16	64	128	64	64	64	0.5660
308	5.49	8	16	32	32	32	64	0.3029
308	7.58	8	32	64	32	32	64	0.3985
308	19.54	8	32	64	32	64	96	0.5321
308	35.36	16	64	128	64	64	64	0.6511
410	5.83	8	16	32	32	32	64	0.3445
410	11.06	8	32	64	32	64	64	0.4648
410	24.72	12	32	96	32	64	64	0.5886
410	37.53	16	64	128	64	64	96	0.6948

FinalMLP

Sparse (B)	FLOPs (G)	mlp1_hidden_units	mlp2_hidden_units	num_heads	RLL (%)
104	2.44	1024	1024	4	-0.9503
104	4.36	2048	2048	8	-0.8241
104	16.98	8192-4096-8192	8192-4096-8192	8	-0.5122
104	36.11	16384-8192-16384	16384-8192-16384	8	-0.4991
206	3.82	1024-1024	1024-1024	4	-1.0278
206	10.00	3072-1536-3072	3072-1536-3072	16	-0.7967
206	13.29	4096-3072-4096	4096-3072-4096	4	-0.7218
206	26.58	8192-4096-8192	8192-4096-8192	8	-0.4786
308	5.03	1024-1024	1024-1024	4	-0.9873
308	9.20	2048-2048	2048-2048	8	-0.7548
308	13.62	3072-1536-3072	3072-1536-3072	4	-0.7998
308	36.17	8192-4096-8192	8192-4096-8192	8	-0.4742
410	6.24	1024	1024	4	-0.9740
410	11.61	2048-2048	2048-2048	8	-0.7246
410	22.80	4096-2048-4096	4096-2048-4096	4	-0.7213
410	34.05	6144-3072-6144	6144-3072-6144	16	-0.6504

AFN

Sparse (B)	FLOPs (G)	logarithmic_neurons	hidden_units	dnn_hidden_units	RLL (%)
104	4.88	16	4096-2048	4096-1024	-1.0631
104	10.90	128	8192-4096-8192	8192-4096-8192	-1.5173
104	15.73	64	12288-6144-12288	12288-6144-12288	-1.3564
104	27.96	128	16384-8192-16384-8192-16384	16384-8192-16384-8192-16384	-1.3583
206	2.41	32	1024-512-1024	1024-512-1024	-1.9392
206	4.15	64	2048-1024-2048	2048-1024-2048	-1.7612
206	7.31	16	4096-2048-4096	4096-2048-4096	-1.4077
206	15.11	64	8192-4096-8192	8192-4096-8192	-1.3615
308	5.40	64	2048-1024-2048	2048-1024-2048	-1.6761
308	9.84	32	4096-2048-4096	4096-2048-4096	-1.4901
308	10.47	128	4096-2048-4096	4096-2048-4096	-1.5382
308	30.21	64	12288-6144-12288	12288-6144-12288	-1.1414
410	6.51	32	2048-1024-2048	2048-1024-2048	-1.6578
410	12.51	64	4096-2048-4096	4096-2048-4096	-1.4501
410	25.55	128	8192-4096-8192	8192-4096-8192	-1.3136
410	37.90	64	12288-6144-12288	12288-6144-12288	-1.1413

DLRM

Sparse (B)	FLOPs (G)	top_mlp_units	compression_num	RLL (%)
104	5.37	2048-4096-2048-4096-2048	–	-0.6936
104	7.28	3072-4096-3072	–	-0.5596
104	17.73	8192-4096-8192-4096-8192	–	-0.4481
104	35.82	16384-8192-16384-8192-16384	–	-0.4684
206	3.50	2048-1024-2048	512	-0.6244
206	8.13	3072-2048-3072	768	-0.5091
206	13.37	2048-1024-2048	–	-0.5815
206	18.77	3072-4096-3072	–	-0.4572
308	8.60	3072-2048-3072	768	-0.4442
308	10.45	4096-2048-4096	768	-0.4022
308	36.02	3072-4096-3072	–	-0.3873
410	6.88	3072-2048-3072	640	-0.4094
410	24.21	1024-512-1024	–	-0.6367
410	46.79	8192-4096-8192	1280	-0.2435
410	59.00	3072-1536-3072	–	-0.4203

DCNv2

Sparse (B)	FLOPs (G)	num_cross_layers	dnn_hidden_units	compression_num	RLL (%)
104	5.40	4	4096-2048-4096	64	-0.4495
104	7.07	6	4096-2048-4096	64	-0.4427
104	16.53	4	4096-2048-4096	128	-0.4048
104	30.99	8	8192-4096-8192	128	-0.3078
206	5.48	4	4096-2048-4096	64	-0.3538
206	7.14	6	4096-2048-4096	64	-0.3723
206	16.63	4	4096-2048-4096	128	-0.3428
206	31.10	8	8192-4096-8192	128	-0.4686
308	5.54	4	4096-2048-4096	64	-0.3887
308	7.71	4	4096-2048-4096	80	-0.3700
308	16.73	4	4096-2048-4096	128	-0.2886
308	31.19	8	8192-4096-8192	128	-0.2658
410	5.60	4	4096-2048-4096	64	-0.3640
410	7.26	6	4096-2048-4096	64	-0.7406
410	16.82	4	4096-2048-4096	128	-0.2500
410	31.29	8	8192-4096-8192	128	-0.5734

FCN

Sparse (B)	FLOPs (G)	exp_num_layers	lin_num_layers	num_heads	output_dim	compression_num	RLL (%)
104	4.38	4	4	2	4096	64	-0.4970
104	7.70	8	8	2	4096	64	-0.6556
104	14.70	16	16	8	8192	128	-1.2395
104	31.84	16	16	8	12288	192	-1.3674
206	4.45	4	4	2	4096	64	-0.4183
206	7.77	8	8	2	4096	64	-0.5774
206	14.80	16	16	8	8192	128	-2.1951
206	31.98	16	16	8	12288	192	-1.6827
308	4.50	4	4	2	4096	64	-0.3630
308	7.83	8	8	2	4096	64	-0.5376
308	14.90	16	16	8	8192	128	-1.9794
308	31.74	16	16	8	8192	192	-1.0432
410	4.57	4	4	2	4096	64	-0.3272
410	7.90	8	8	2	4096	64	-0.4896
410	15.00	16	16	8	8192	128	-1.5484
410	31.87	16	16	8	8192	192	-0.8979

APG

Sparse (B)	FLOPs (G)	hidden_units	rank_k	RLL (%)
104	2.46	1024-1024	32	-0.8465
104	6.17	2048-2048	64	-0.6558
104	10.85	8192-4096-8192	48	-1.7752
104	19.23	4096-4096	128	-0.8598
206	5.29	2048-1024-2048	32	-1.3099
206	9.85	2048-1024-2048	64	-1.7797
206	19.79	8192-4096-8192	64	-2.1590
206	27.65	8192-4096-8192	96	-2.2536
308	4.94	1024-1024	32	-0.9628
308	11.53	4096-2048-4096	32	-1.3794
308	23.15	8192-8192	48	-0.7002
308	27.03	8192-4096-8192	64	-1.4153
410	4.12	1024-512-1024	16	-1.4079
410	8.96	2048-1024-2048	32	-1.3862
410	17.15	2048-1024-2048	64	-2.1948
410	34.16	8192-8192	64	-0.7431

MaskNet

Sparse (B)	FLOPs (G)	hidden_units	num_blocks	block_dim	compression_num	RLL (%)
104	2.56	3072-1536-3072	4	128	32	-0.6262
104	6.70	4096-2048-4096	6	192	48	-0.4916
104	14.28	2048-1024-2048	8	256	64	-0.2766
104	28.42	4096-2048-4096	16	512	64	-0.1806
206	2.62	3072-2048-3072	4	128	32	-0.5648
206	6.75	4096-2048-4096	6	192	48	-0.4238
206	14.34	2048-1024-2048	8	256	64	-0.1998
206	28.48	4096-2048-4096	16	512	64	-0.2670
308	2.67	3072-2048-3072	4	128	32	-0.5190
308	8.75	4096-2048-4096	8	256	48	-0.1916
308	14.32	2048-1024-2048	8	128	64	-0.2083
308	28.53	4096-2048-4096	16	512	64	-0.0580
410	2.15	4096-2048-4096-2048-4096	4	128	24	-0.4839
410	6.86	4096-2048-4096	6	192	48	-0.3571
410	14.38	2048-1024-2048	8	128	64	-0.1654
410	41.43	4096-2048-4096	6	128	128	-0.0624

RankMixer

Sparse (B)	FLOPs (G)	T	D	L	RLL (%)
104	5.89	64	512	6	-0.1140
104	15.79	64	768	8	0.0415
104	23.04	96	768	8	0.0805
104	44.78	96	768	16	0.1330
206	6.21	64	512	6	-0.1057
206	16.26	64	768	8	0.0305
206	23.51	96	768	8	0.0984
206	34.39	96	768	12	0.1120
308	6.54	64	512	6	-0.1223
308	16.74	64	768	8	-0.1331
308	23.98	96	768	8	0.0951
308	34.86	96	768	12	0.1314
410	6.86	64	512	6	0.0064
410	24.46	96	768	8	0.1227
410	35.33	96	768	12	0.1874
410	42.06	64	1024	12	0.1564

The DHEN and Maglev-DHEN grids span the same Rank range (32–64). Maglev-Wukong uses Rank 96 in all 8 of its configurations above 15 GFLOPs, whereas Wukong's grid spans 32–128. Restricted to Rank ≤ 64 (8 configurations), Maglev-Wukong still lies above the Wukong frontier in all 24 configuration–task comparisons that have one. The two Wukong pairs that differ only in Rank (16/128/128; Rank 64 vs. 128 at 206B and 32 vs. 128 at 410B, at 32–51% more compute) differ by -0.05 to $+0.15$ pp across tasks, with mixed sign.

A.6.3 Depth Scaling Laws

Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	RLL (%) Improv.
104	6.15	1	128	128	128	0.0000
104	6.39	2	96	96	64	0.0540
104	6.18	4	64	64	64	0.0980
104	6.12	8	72	72	32	0.1410
104	6.16	16	32	32	28	0.1610
104	6.21	32	24	24	24	0.0870

Maglev-Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	Indexers	RLL (%) Improv.
104	6.18	1	0	256	128	128	-0.0030
104	5.50	2	0	256	128	128	0.0520
104	6.03	4	0	64	96	192	0.1640
104	6.57	8	0	128	128	32	0.2240
104	6.17	16	16	32	32	32	0.2410
104	6.54	32	32	32	32	16	0.2510

A.6.4 Dense Scaling Laws

	Dense (B)	Sparse (B)	FLOPs (G)	Dense width†	L	RLL (%) Improv.
Wukong	2.2	135	14.8	3072-2048	10	-0.00
	5.1	135	32.1	6144-4096	10	0.13
	6.8	135	42.2	7680-5120	10	0.11
	12.3	135	75.5	12288-8192	10	0.12
	17.8	135	108.3	15360-10240	10	0.13

Maglev-Wukong

Dense (B)	Sparse (B)	FLOPs (G)	Dense width†	L	RLL (%) Improv.
2.1	135	14.4	3072-2048	10	0.10
4.8	135	30.7	6144-4096	10	0.17
6.4	135	40.3	7680-5120	10	0.21
12.3	135	75.5	12288-8192	10	0.25
17.0	135	104.0	15360-10240	10	0.27

†MLP widths. Other internal widths are co-tuned at each budget and are not listed.

A.6.5 SPARSE SCALING LAWS

From 104B to 135B, new features are added; from 135B, embeddings are widened: those of the 50, 100, and 200 most important fields (by feature-selection importance) at 149, 191, and 300B, and of every field at 514B. Both arms follow the same schedule.

Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	Dense width†	RLL (%) Improv.
104	29.97	10	64	80	128	3648-5472	0.0000
135	30.19	10	64	80	128	3120-4680	0.2850
149	30.42	10	64	80	128	2880-4320	0.3400
191	30.16	10	64	80	128	2464-3680	0.3840
300	31.11	10	64	80	128	1536-3072	0.4000
514	32.36	10	64	80	128	512-768	0.4300

Maglev-Wukong

Sparse (B)	FLOPs (G)	L	LCE	FM	Rank	Indexers	Dense width†	RLL (%) Improv.
104	30.69	10	64	80	128	64	4112-6168	0.0600
135	30.64	10	64	80	128	64	4112-6168	0.3270
149	30.78	10	64	80	128	64	4112-6168	0.3810
191	30.91	10	64	80	128	96	4112-6168	0.3990
300	31.20	10	64	80	128	96	3696-5544	0.5500
514	30.63	10	64	80	128	96	3280-4920	0.5900

†MLP widths.

A.7 MAGLEV ADAPTATION TO SEQUENCE FEATURES

The industrial study includes 100 sequence features, so this mechanism is exercised in the reported industrial results. Sequence features enter as unpooled sparse embeddings. For Hard Maglev Indexers, each sequence's temporal axis is divided into ordered chunks assigned across the participating stages; Each stage applies the backbone's existing sequence interaction to its local chunk, while Maglev Induct propagates context forward so that later stages condition on earlier subsequences.

A.8 REFERENCE IMPLEMENTATION

For reproducibility, we provide a reference Maglev implementation in PyTorch-style pseudocode (Listing 1). Three components are shown:

- `MaglevHardIndexer` realizes Maglev Indexers by assigning $S_i = |\texttt{partitions}[i]|$ raw feature fields to stage i and then compressing them to $P_i = \texttt{out_sizes}[i]$ outputs.
- `InductGate` realizes the Maglev Induct gating function

$$R' = \mathrm{LN}(\lambda R + \mu\,(YW_y \odot \sigma(XW_x)))\,.$$

  : W_y maps Y to the Induct width O, and $\sigma(XW_x)$ gates it using the previous stage's dense activations.
- `MaglevModel` wraps an arbitrary sequence of recommendation stages and routes the per-stage indexer outputs and the Induct feature bus through them.

`LinearCompress` is the underlying linear projection along the feature axis, used for both the indexer compression and `InductGate`'s W_y/W_x projections. Boilerplate (configuration, type annotations, distributed-training hooks) is omitted; the pseudocode is faithful to the paper description up to those omissions. Maglev Rail's pipeline-parallel execution of these stages is orthogonal to this listing and lives in the framework layer.

The listing begins after distributed stage-local embedding lookup. Accordingly, `stage_embeddings[i]` contains lookup outputs from stage i's independently parame-

terized tables; a field assigned to multiple stages therefore produces distinct embedding tensors. Raw-input routing and distributed table sharding reside in the framework layer.

Listing 1: Maglev reference implementation. `MaglevModel` wraps an arbitrary sequence of recommendation stages with a `MaglevHardIndexer` (per-stage feature partition + linear compression) and Maglev Induct (low-rank residual feature bus R updated as $R' = \mathrm{LN}(\lambda R + \mu\,(YW_y \odot \sigma(XW_x)))$ across stages).

```
import torch
import torch.nn as nn


# ============================================================
# Maglev Indexers: per-stage feature partition + compression
# ============================================================
class MaglevHardIndexer(nn.Module):
    # Permutes input features by stage assignment, then linearly
      compresses
    # each partition to its target output size. Realizes Hard Indexers
      with
    # optional Soft compression within each partition.
    def __init__(self, partitions, out_sizes, emb_dim):
      # partitions: List[List[int]], stored feature indices; len(
        partitions[i]) = S_i
      # out_sizes:  List[int], compressed output widths; out_sizes[i] =
        P_i
        super().__init__()
        self.in_sizes    = [len(p) for p in partitions]
        self.out_sizes   = out_sizes
        self.compressors = nn.ModuleList([
            LinearCompress(self.in_sizes[i], out_sizes[i], emb_dim)
            if out_sizes[i] != self.in_sizes[i] else nn.Identity()
            for i in range(len(partitions))
        ])

  def forward(self, stage_embeddings):
      # stage_embeddings[i]: (B, S_i, D), produced by stage-i tables
      return [
          self.compressors[i](stage_embeddings[i])
          for i in range(len(stage_embeddings))
      ]


# ============================================================
# Maglev Induct Gate: R' = LN(lam * R + mu * (Y W_y (*) sigmoid(X W_x)))
#                                              ^^^ elementwise (Hadamard)
# ============================================================
class InductGate(nn.Module):
  # Inputs:
  #   R: prior residual digest, shape (B, O, D), or None at the first
      call
  #   Y: current stage's indexer output, shape (B, P, D)
  #   X: previous stage's dense activations, shape (B, F_X, D)
  # Internals:
  #   W_y: linear projection (P -> O) aligning Y to the bottleneck dim O
  #   W_x: linear projection (F_X -> O) so sigmoid(X W_x) matches Y W_y
  #   lam, mu: scalar mixing coefficients
  def __init__(self, y_features, induct_dim, x_features, emb_dim):
      super().__init__()
      self.lam  = nn.Parameter(torch.ones(1))
      self.mu   = nn.Parameter(torch.ones(1))
      self.W_y  = LinearCompress(y_features, induct_dim, emb_dim)
      self.W_x  = LinearCompress(x_features, induct_dim, emb_dim)
      self.norm = nn.LayerNorm(emb_dim)
```

```
  def forward(self, r, y, x):
      yw    = self.W_y(y)                                  # (B, O, D)
      gated = yw * torch.sigmoid(self.W_x(x))              # Y W_y (*)
          sigmoid(X W_x)
      if r is None:
          return self.norm(self.mu * gated)
      return self.norm(self.lam * r + self.mu * gated)


# ============================================================
# Maglev Model: wraps an arbitrary sequence of stages
# ============================================================
class MaglevModel(nn.Module):
    # Wraps a sequence of arbitrary recommendation stages with Maglev
        Indexers
    # (per-stage feature partition + compression) and Maglev Induct (low-
        rank
    # residual feature bus carried across stages).
    def __init__(self, stages, indexer, induct_dim, stage_features,
        emb_dim):
        # stages:         nn.ModuleList of arbitrary stage modules
        # indexer:        MaglevHardIndexer
        # induct_dim:     residual bus dimension (O in the paper)
        # stage_features: feature count of each stage's dense output (X)
        super().__init__()
        self.stages  = stages
        self.indexer = indexer
        K = len(stages)
          self.induct_gates = nn.ModuleList([
              InductGate(indexer.out_sizes[i + 1], induct_dim,
                         stage_features[i], emb_dim)
              for i in range(K - 1)
          ])

      def forward(self, stage_embeddings, **stage_kwargs):
          segments = self.indexer(stage_embeddings)

          prev_out = self.stages[0](
              segments[0], **stage_kwargs
          )
          induct_bus = None

          for i in range(1, len(self.stages)):
              induct_bus = self.induct_gates[i - 1](
                  induct_bus, segments[i], prev_out
              )
              stage_in = cat_nonnone(
                  prev_out, segments[i], induct_bus
              )
              prev_out = self.stages[i](
                  stage_in, **stage_kwargs
              )

          return prev_out


# ============================================================
# Helpers
# ============================================================
def cat_nonnone(*tensors):
    # Concatenate non-None tensors along the feature axis.
    return torch.cat([t for t in tensors if t is not None], dim=1)


class LinearCompress(nn.Module):
```

```
    # Linear projection along the feature axis: (B, F_in, D) -> (B, F_out
        , D).
    def __init__(self, in_features, out_features, emb_dim):
        super().__init__()
        self.weight = nn.Parameter(torch.empty(out_features, in_features)
            )
        nn.init.xavier_uniform_(self.weight)

    def forward(self, x):
        return torch.einsum('fg,bgd->bfd', self.weight, x)
```

A.9 Full Analysis and Proof of the Maglev Induct Representation Bound

For fixed B_i, minimizing $\|W - B_i U_i\|_F^2$ is a linear least-squares problem. One optimal solution is $U_i^\star = B_i^\dagger W$, giving

$$B_i U_i^\star = B_i B_i^\dagger W = \Pi_i W.$$

Because $B_i = Q_i B_i$, its column space is contained in the range of Q_i. Hence $Q_i \Pi_i = \Pi_i Q_i = \Pi_i$, and

$$I - \Pi_i = (I - Q_i) + (Q_i - \Pi_i).$$

The two summands project onto orthogonal subspaces:

$$(I - Q_i)(Q_i - \Pi_i) = 0.$$

The Pythagorean identity therefore gives

$$\|(I - \Pi_i)W\|_F^2 = \|(I - Q_i)W\|_F^2 + \|(Q_i - \Pi_i)W\|_F^2 .$$

Finally,

$$\|z(\mathbf{f})(W - \Pi_i W)\|_2 \le \|z(\mathbf{f})\|_2 \, \|(I - \Pi_i)W\|_2 \le \|z(\mathbf{f})\|_2 \, \|(I - \Pi_i)W\|_F .$$

which proves Equation 1.

Since $\mathrm{rank}(B_i) \le O$, the Eckart–Young–Mirsky theorem gives

$$\|(Q_i - \Pi_i)W\|_F^2 \ge \sum_{k>O} \sigma_k^2(Q_i W),$$

with equality when $\mathrm{col}(B_i)$ spans the leading O left-singular directions of $Q_i W$. At the final stage, full feature coverage gives $Q_K = I$; hence the causal-deferral term vanishes, while exact recovery additionally requires the effective interaction rank to be at most O.

Relation to the Deployed Gated Block The deployed update (§3.2) is

$$R' = \mathrm{LN}(\lambda R + \mu \, (Y W_y \odot \sigma(X W_x))) .$$

Gating strictly enlarges the function class of the ungated residual update. With $W_x = 0$ the gate is the constant $1/2$, which recovers the ungated update up to the scale of β; with $W_x \neq 0$ the retained subspace depends on X_i. At optimal parameters, gating therefore cannot increase the ungated block's representation error, and the rank-O floor, which assumes a fixed linear factor B_i, need not bind.

On a bounded activation domain, finite-norm linear projections, sigmoid gating, bounded Hadamard products, LayerNorm with nonzero stabilization, and the stage MLPs form a Lipschitz map. Thus, if Φ_i denotes the downstream deployed computation with Lipschitz constant L_i,

$$\|\Phi_i(h_\star) - \Phi_i(\widehat{h}_i)\|_2 \le L_i \|h_\star - \widehat{h}_i\|_2.$$

This transfers the surrogate bound up to a generally non-tight constant; it does not guarantee that nonlinear training attains the optimal factorization or bound the full model's prediction error.

A.10 Detailed Experimental Setup

Baselines We compare Maglev against 10 recommenders: Wukong (Zhang et al., 2024a), DCNv2 (Wang et al., 2021a), MaskNet (Wang et al., 2021b), AFN (Cheng et al., 2020), DLRM (Naumov et al., 2019), FinalMLP (Mao et al., 2023), DHEN (Zhang et al., 2022; Zhuang et al., 2025), FCN (Li et al., 2025), APG (Yan et al., 2022) and RankMixer (Zhu et al., 2025b). We used identical batch size and optimizer settings and proper warmup schedule for all models. We transform RankMixer, FinalMLP, AutoInt, DESTINE, and FiGNN for public dataset comparison, and Wukong and DHEN for the large-scale industrial study. Wukong and DHEN were not selected in advance: they reached the highest untransformed RLL at every sparse capacity and task in the 10-family industrial screen and are widely used by the industry (Zhuang et al., 2025; Luo et al., 2025; Cai et al., 2026; Zeng et al., 2024; Hou et al., 2026; Luo et al., 2026). They also represent different interaction structures: Wukong repeatedly stacks specialized interaction blocks, whereas DHEN recursively ensembles heterogeneous interaction modules. The remaining eight families serve as untransformed comparison points in the scaling plots.

Datasets Our primary scaling-law study uses a 28-billion-sample, 700-feature industrial multi-domain, multi-objective dataset comprising 600 sparse features and 100 sequence features. Standard public recommendation benchmarks are too small and saturated to reproduce this industrial scaling regime (Zhang et al., 2024a; 2023; Zeng et al., 2024; Zhu et al., 2025b; Xu et al., 2025b; Shin et al., 2023; Yan et al., 2026; Zhai et al., 2024; Hou et al., 2026). Nevertheless, they provide reproducible external checks of relative model behavior. We therefore additionally evaluate Maglev on KKBox (kkb), Criteo (Dis), and KuaiVideo (Kuaishou). Criteo is the 45.8M-sample release distributed with FuxiCTR/BARS, split sequentially 72/18/10 into training, validation, and test; KKBox and KuaiVideo use the BARS Zhu et al. (2022) splits. Batch size is fixed per dataset for both arms (4,096 for Criteo and KKBox, 8,192 for KuaiVideo), and every model is held to 0.2–5 MFLOPs, so absolute AUCs are not comparable with published BARS results; comparisons are paired within each budget. We use the public experiments as matched-compute sanity checks rather than as substitutes for the industrial scaling-law study. Our experiment is conducted on a heterogeneous cluster with thousands of GPUs. The finished jobs took roughly 800K H100-normalized hours. We normalize device-hours using coarse peak-compute weights: one A100-hour equals 0.5 H100-hours, while one B200/GB200/MI350X-hour equals two H100-hours. The reported total excludes retries and is intended as an approximate compute-accounting measure.

Metrics and Uncertainty We report Relative LogLoss (RLL) improvement over a fixed baseline configuration, retrained on the same data as each candidate, $\mathrm{RLL}(X) = 100\% \cdot (\mathrm{LogLoss}(\mathrm{baseline}) - \mathrm{LogLoss}(X))/\mathrm{LogLoss}(\mathrm{baseline})$ where higher is better. Each large-scale industrial configuration was executed once under a common fixed-seed policy due to excessive resource cost. The 0.01% RLL threshold is the production platform's fixed-configuration A/A reproducibility tolerance, established from thousands of historical reruns of identical configurations; it is not an estimate of variation across random seeds. We therefore interpret improvements above 0.01% as exceeding the platform's reproducibility tolerance and as operationally meaningful, rather than as evidence of generic statistical significance. Whether a single run resolves such a difference depends on seed variation (§A.10.1), which the reported effects exceed. The public experiments provide complementary multi-seed evidence through 12 matched seeds per setting, paired estimates, and 95% confidence intervals. For matched seed s, we define $d_s = m_s^{+\mathrm{Maglev}} - m_s^{\mathrm{base}}$ and report the two-sided 95% Student-t interval $\bar{d} \pm t_{0.975,n-1} s_d/\sqrt{n}$. We report the primary CTR task in the main scaling plots and additional tasks in an aggregated manner in Table 2.

A.10.1 Fixed-Configuration Repeatability Check on Later Data

We separately repeated one fixed industrial contrast using seeds 0–4 on both A100 and GB200 hardware. The baseline is a 104B-sparse, 3.47-GFLOP, four-layer Wukong configuration; the candidate is a 104B-sparse, 4.81-GFLOP, eight-stage Maglev-DHEN configuration. We report only paired candidate-minus-baseline RLL deltas, not absolute RLL levels. For seed-level inference, we average the two hardware replicates for each seed and form a Student-t interval over the five seed averages. All 40 task–seed–hardware deltas are positive (Table 7). These runs use a later data window than the scaling campaign, yet the campaign's run of the same contrast lies inside the interval on all four tasks (within 0.008 pp of the mean), so RLL measured against a baseline trained on the same data transfers across data windows.

Table 7: Fixed-configuration repeatability. Values are RLL percentage-point deltas. A100 and GB200 columns are means over five seeds; the final mean, SD, and CI treat seed as the statistical unit after averaging the two hardware replicas.

Task	A100 mean	GB200 mean	Mean Δ	Seed SD	95% CI
CTR Task 1	0.1396	0.1324	0.1360	0.0125	[0.1205, 0.1516]
CVR Task 1	0.1540	0.1561	0.1550	0.0108	[0.1417, 0.1684]
CVR Task 2	0.1435	0.1360	0.1398	0.0132	[0.1234, 0.1561]
CVR Task 3	0.1391	0.1317	0.1354	0.0118	[0.1207, 0.1501]

Because these fixed configurations differ in backbone and by 38.6% in forward FLOPs, this check characterizes repeatability of the observed RLL improvements only. It is not used as an iso-FLOP efficacy comparison, a component-attribution experiment, or a seed-uncertainty estimate for the frontier comparison.

Why results are not pooled across public settings. Because the datasets use different metrics and the backbone–dataset settings are correlated design points, we report paired inference within each setting rather than pooling raw deltas across settings. At matched compute, Maglev yields a positive mean quality delta in all 15 evaluated settings; 14 nominal paired 95% intervals exclude zero, and the same 14 remain after Holm correction at family-wise level 0.05.

A.11 Frontier Comparison Details

Frontier comparison. For each arm and sparse capacity P, the baseline frontier is the running maximum of RLL over that arm's configurations ordered by FLOPs G, including the Wukong normalization anchor ($P = 104$B, $G = 3.47$ GFLOPs, RLL $= 0$ by construction). We interpolate it linearly in $\log G$ between consecutive frontier configurations, hold it flat beyond the most expensive one, and do not extrapolate below the cheapest. Three Maglev configurations (Maglev-Wukong at 308B, 4.06 GFLOPs and 410B, 4.45 GFLOPs; Maglev-DHEN at 308B, 5.49 GFLOPs) are cheaper than every same-backbone baseline configuration at their capacity, so 116 of the 128 configuration–task pairs have a comparator. Compute to match is the FLOPs at which the frontier first reaches the Maglev configuration's RLL, divided by that configuration's FLOPs. Without interpolation, comparing with the best baseline configuration at equal or lower G also gives 116 of 116 wins, with a median margin of 0.22 pp; that comparator uses a median $1.36\times$ less compute than the Maglev configuration, a gap that interpolation removes. Including every converged Wukong, DHEN, and Maglev run we trained (98 configurations) gives 152 of 152 wins (median margin 0.12 pp).

Matched-budget check. As a check that needs no interpolation, we pair each Maglev configuration with the nearest same-backbone baseline configuration at identical sparse capacity, retaining pairs with $|G_{\text{Maglev}}/G_{\text{base}} - 1| \leq \tau$. At $\tau = 5\%$ this yields eight pairs, and every pair favors Maglev on all tasks, as do all four pairs at $\tau = 3\%$ and all 15 at $\tau = 10\%$. These are descriptive checks across deliberately selected configurations, not IID replications or inputs to a significance test.

Isolated sweeps. For the isolated sweeps, deleting each corresponding scale point from both arms preserves every direction; original-fit R^2 is $0.64/0.57/0.53$ (dense/sparse/depth) for Wukong and $0.98/0.80/0.90$ for Maglev.

A.12 Detailed Ablation

A.12.1 End-to-end Ablation

We first provide an end-to-end ablation on a 4-layer Wukong model in Figure 4 (a). Starting with naive feature disaggregation, we observe quality degradation of $0.02 - 0.05\%$; applying Maglev Indexers recovers the most important feature interactions (capacity $= 128$) and results in 0.04% RLL improvement; replacing it with the deployable Hard Indexer alone lowers the gain to 0.01%, which is necessary to enable Rail execution, and adding Maglev Induct ($O = 32$) raises it to 0.05%.

A.12.2 EFFECTIVENESS OF MAGLEV INDEXERS

Partition Strategies We compare Maglev Indexers's learned partitioning against two strategies: random assignment and importance-based selection (Debnath et al., 2008; Jain et al., 2016). We convert a 4-layer Wukong model into an iso-FLOP, iso-sparse 4-stage Maglev model. For the importance-based variant, we test three distributions: *Ascending* and *Descending* (features sorted by importance, chunked sequentially) and *Interleaved* (round-robin striped across stages, distributing high-value features evenly).

Figure 4 (a) shows a counterintuitive result: every importance heuristic underperforms random partitioning. Static importance fails to generalize to disaggregated settings — only Maglev Indexers's learned, affinity-based co-location beats random, by an additional $+0.04\%$ RLL.

Indexer Output Capacity. The output width P_i controls how many compressed feature combinations stage i receives; it is distinct from the raw-field storage budget S_i. On an 8-stage Maglev-Wukong model, quality improves as P_i increases from 0 to 128 and saturates near 64 (Figure 4(b)). In this setting, $P_i \approx F/(2K)$ provides sufficient compressed output bandwidth. Global feature coverage is governed separately by the storage budgets, which satisfy $\sum_i S_i \geq F$.

Field Reuse. On a 10-stage Maglev model trained on a separate production dataset with $F = 1{,}645$ feature fields, we raise $\sum_i S_i$ from F (each field assigned once) to $2.01F$ at fixed P_i, Maglev Induct width, and backbone. The input service fans out the same raw IDs or values to every stage that holds a reused field, but each stage learns its own table independently; no embedding parameters or gradients are synchronized across copies. Sparse parameters therefore grow in proportion (to about $2\times$), while forward FLOPs grow by only 1.0%. RLL on the primary task changes by at most 0.03 pp: modest reuse (1.1–$1.3F$) scores highest, and $2.01F$ matches $1.00F$. Coverage with little reuse therefore suffices: beyond about 1.3F, placing a field in more stages brings no gain, unlike widening its embedding within one stage (§4.3). At $\sum_i S_i = F$, the Hard-Indexer model also improves RLL by 0.07 pp over a Soft-Indexer model with the same backbone and sparse parameters (Indexer output 160 rather than 144, no Maglev Induct), while using 6% fewer FLOPs.

Hard Indexers on public benchmarks. We replace the Soft Indexer in each public setting with a Hard Indexer, so each stage reads only its assigned fields, and evaluate it against the same native baselines and 12 matched seeds as Table 1 (Table 8). The Hard Indexer improves on the native backbone in 14 of 15 settings, and in 13 the paired 95% interval excludes zero. Against the Soft Indexer the results are mixed: Hard is higher in 9 settings, including every FinalMLP and AutoInt setting, and lower in 6, including all three FiGNN settings. Confining each stage to its own fields, as required once stages span several HSDs, therefore keeps Maglev's gain over the native backbone.

§A.5, $\mathcal{A}$ stabilizes early in training, keeping feature-to-stage assignments coherent throughout the model's lifecycle.

A.12.3 EFFICACY OF MAGLEV INDUCT

Induct Bottleneck Width (O) We isolate Maglev Induct's conditional representation learning by varying the residual bottleneck dimension O on an 8-stage Maglev model and reporting the quality–complexity trade-off relative to a maximum-capacity $O = 128$ baseline. Figure 4 (c) shows RLL improvements saturating quickly: Maglev Induct compresses and recovers global interactions without needing full-width dimensionality. The optimal trade-off emerges around $O = 32$, yielding a 14% reduction in interaction complexity with negligible quality loss.

Feature Ensemble Mechanisms We compare additive summation (followed by LayerNorm) against tensor concatenation for fusing the residual R' with prior-stage outputs, with and without gating. Differences are small: summation vs concatenation converge within 0.02% RLL of each other; gating contributes an additional 0.03% RLL improvement.

A.12.4 OPTIMAL SPARSE SCALING STRATEGY

We compare two approaches to sparse scaling: (1) *Horizontal* scaling, which introduces entirely new sparse features, and (2) *Vertical* scaling, which widens the embeddings of existing features. For vertical scaling, we test two selection rules: only the most critical features (by importance heuristics)

Table 8: Hard Indexers on public benchmarks. Native and Soft are the 12-seed means from Table 1; Hard replaces the Soft Indexer with a Hard Indexer. Δ_{H-N} uses the same 12 matched seeds with nominal paired 95% CIs; $H - S$ is a difference of means. Bold marks the higher of Soft and Hard. KKBox and Criteo report AUC; KuaiVideo reports gAUC.

Backbone	Dataset	Native	Soft	Hard	Δ_{H-N} [95% CI]	$H - S$
RankMixer	KKBox	0.815562	0.818318	**0.831936**	+0.016374 [0.015864,0.016884]	+0.013618
	Criteo	0.810631	**0.811749**	0.811285	+0.000653 [0.000360,0.000947]	-0.000464
	KuaiVideo	0.648478	**0.649948**	0.649205	+0.000727 [0.000112,0.001342]	-0.000743
FinalMLP	KKBox	0.813938	0.814895	**0.815863**	+0.001925 [0.001038,0.002811]	+0.000968
	Criteo	0.806013	0.806320	**0.806327**	+0.000314 [0.000208,0.000419]	+0.000007
	KuaiVideo	0.635644	0.639249	**0.641344**	+0.005700 [0.004494,0.006906]	+0.002095
AutoInt	KKBox	0.810655	0.813334	**0.813510**	+0.002855 [0.001520,0.004191]	+0.000176
	Criteo	0.802731	0.804484	**0.804981**	+0.002250 [0.001967,0.002534]	+0.000497
	KuaiVideo	0.632544	0.633641	**0.638364**	+0.005820 [0.004486,0.007154]	+0.004723
DESTINE	KKBox	0.804340	**0.812475**	0.812344	+0.008004 [0.003761,0.012248]	-0.000131
	Criteo	0.803869	0.804427	**0.806167**	+0.002298 [0.002107,0.002488]	+0.001740
	KuaiVideo	0.626969	0.632760	**0.634997**	+0.008028 [0.006305,0.009751]	+0.002237
FiGNN	KKBox	0.808652	**0.816733**	0.811749	+0.003096 [0.002197,0.003996]	-0.004984
	Criteo	0.803982	**0.804934**	0.803942	-0.000040 [-0.000429,0.000349]	-0.000992
	KuaiVideo	0.628795	**0.633893**	0.629680	+0.000885 [-0.000248,0.002018]	-0.004213

versus a random subset. We additionally test whether scaling the dense continuous inputs provides compounding benefits.

Figure 4 (d) shows horizontal scaling delivers the highest absolute quality but requires manual feature engineering to source new signals. Vertical scaling is a pragmatic alternative: competitive gains purely through algorithmic capacity expansion, with zero data engineering overhead.

A.12.5 Throughput Impact of Maglev Rail

We benchmark Maglev Rail against the ZB2P implementation in Megatron-LM (Narayanan et al., 2021) while enforcing synchronous gradient clipping. On a 16-stage, 64-GFLOP Maglev model running on 128 A100 GPUs, the complete system delivers up to $1.3\times$ end-to-end throughput relative to the original production training system. In a sequential component ablation, the zero-bubble schedule contributes $1.10\times$, PPTwCw sharding contributes a further $1.10\times$, and tuned computation–communication overlap contributes a further $1.07\times$, yielding $1.29\times$ cumulatively.

Both Maglev Rail and ZB2P use $M = P$, the same global batch size, stage partition, clipping threshold, and optimizer configuration. For ZB2P, all current-iteration weight gradients are drained before the global norm reduction and clipping barrier; no gradient computation is moved across iterations. Reported throughput includes gradient-norm reduction, clipping, and the optimizer step for both schedules.

To isolate the schedule itself, we additionally compare Maglev Rail with ZB2P across 12 measured configurations spanning 13B, 24B, and 64B models; 128 and 256 A100 GPUs; and pipeline depths of 16 and 32. Maglev Rail improves measured throughput by 6.3%–11.7% in every configuration (§ A.4). These are real-system measurements and therefore include ordinary runtime and topology variation rather than being projections from the equal-latency analysis.

A.13 Additional Related Work

Scaling Laws Existing work on recommendation model scaling laws largely study isolated dense scaling (Zhang et al., 2024a; Hou et al., 2026; Zeng et al., 2024; Zhai et al., 2024; Xu et al., 2025b; Zhang et al., 2024b; Shin et al., 2023; Yan et al., 2026) and sparse scaling (including feature scaling, sequence scaling) (Xu et al., 2025b; Yan et al., 2026). Additionally, Ardalani et al. (2022) jointly fit embedding and dense parameters for DLRM; Maglev differs in redesigning interaction so that added

sparse capacity keeps improving quality, and in scaling depth at fixed compute. Maglev bridges the gap in the literature.

Affinity-based Partition DMT (Luo et al., 2024) clusters feature fields using embedding dot-product similarity and constrained K-Means (Bradley et al., 2000) to construct balanced horizontal partitions. Maglev instead learns affinities under explicit per-stage storage and output-bandwidth budgets and uses them to construct sequential stage-local inputs. Compared with MoE routers (Gale et al., 2022; Shazeer et al., 2017), hard Indexers maintain global field coverage, while soft Indexers learn continuous field combinations rather than token-level discrete routing.

Learned feature grouping (Liu et al., 2020a) and interaction selection (Liu et al., 2020b) decide which feature interactions to model, and RankMixer groups features into tokens by semantics (Zhu et al., 2025b); these operate within a single monolithic model, whereas Maglev Indexers's partition determines which stage, and hence which scale-up domain, stores and processes each field, under explicit storage and bandwidth budgets.

Context Augmentation and Conditional Learning Qwen (Qiu et al., 2025) leverages gating to extract contexts from hidden latents; DeepSeek Engram (Cheng et al., 2026) gates additional N-gram embedding to apply external knowledge to the current context; Gemma (Yu et al., 2025) adds a per-layer embedding, although Maglev uses similar per-stage embedding, and multi-embedding designs (Guo et al., 2023) learn several tables per field to counter embedding collapse. Maglev's stage-local tables are related: a field reused by several stages receives an independent table in each. Unlike these designs, reuse in Maglev is bounded by per-stage storage budgets and decided by the Indexer. mHC (Xie et al., 2026), HC (Zhu et al., 2025a), and DenseNet (Huang et al., 2018) improve context flow via wider and denser connectivity, incurring additional projection overheads.

Linear-complexity Interaction Methods While successful in language modeling, linear methods (Gu & Dao, 2024; Dao & Gu, 2024; Yang et al., 2025; Team et al., 2025), sparse attention (DeepSeek-AI et al., 2025; Child et al., 2019), linear attention (Choromanski et al., 2022; Wang et al., 2020), windowed attention (Fu et al., 2025) and hybrid methods (Bae et al., 2026) are yet to be adopted by recommenders, mostly due to the diverse input format (mixed sequence, nonsequence, and dense features).

Composite Industrial Recommenders Meta Lattice (Luo et al., 2025) and Kunlun (Hou et al., 2026) are composite recommendation systems rather than atomic interaction backbones. Meta Lattice composes DHEN and Wukong Fusion Blocks, while Kunlun's global interaction module uses a mixture of Wukong experts. At these block boundaries, a Maglev transformation would assign Fusion Blocks or expert groups to stage-local computation, use Maglev Indexers for learned feature partitioning, and use Maglev Induct to propagate cross-stage context. We do not report head-to-head comparisons against their complete systems, because such comparisons would combine the proposed architectural transformation with orthogonal data-sharing, distillation, sequence-modeling, and systems components; nor do we claim that Maglev subsumes either complete system.

Limitations The industrial scaling study uses proprietary data and one fixed seed per configuration because the full sweep consumed approximately 800K H100-normalized GPU-hours. A contemporary fixed-configuration check measures seed repeatability at one contrast but does not provide seed-level uncertainty for the complete scaling grid. We complement the industrial study with paired-seed results on three public benchmarks, although these smaller datasets do not reproduce the industrial scaling regime.

The Maglev Induct bound applies to an additive linear surrogate, while the Maglev Rail structural zero-bubble guarantee assumes equal scheduled operation latency; our empirical throughput results measure real execution across 12 configurations. Accordingly, our claims are limited to the evaluated empirical settings and the stated analytical assumptions.

A.14 Statistical Units and Scope of Inference

Industrial scaling comparisons. Configurations in the scaling sweep are deliberately selected design points, not IID samples from a population of configurations, so a configuration bootstrap would impose an artificial exchangeability assumption. Likewise, the historical A/A tolerance is an operational threshold rather than a matched estimate of run-to-run noise, so we do not use it as a noise distribution. Because the sweep uses one seed per configuration, we report no seed-level confidence

intervals; instead, §A.11 repeats the comparison without interpolation, with every converged run, and at matched budgets. The five-seed, two-hardware check (§A.10.1) concerns one fixed contrast and is not propagated to the grid.

Public benchmarks. Each public cell estimates a paired treatment effect for one fixed dataset, metric, backbone, and validation-selected configuration. The cells mix AUC and gAUC and are correlated design points rather than exchangeable replications of a single population effect. Pooling their raw deltas, or treating the three datasets and five backbones as random samples from a superpopulation, would make the result depend primarily on an unverifiable normalization and random-effects model. We therefore report paired inference within each cell. All 15 point estimates are positive, and 14 nominal cell-wise 95% intervals exclude zero. These intervals summarize estimation uncertainty within each setting rather than simultaneous coverage. Applying Holm's step-down correction to the 15 paired Student-t tests retains the same 14 settings at family-wise level 0.05. We do not treat the cells as exchangeable replications of one population effect.

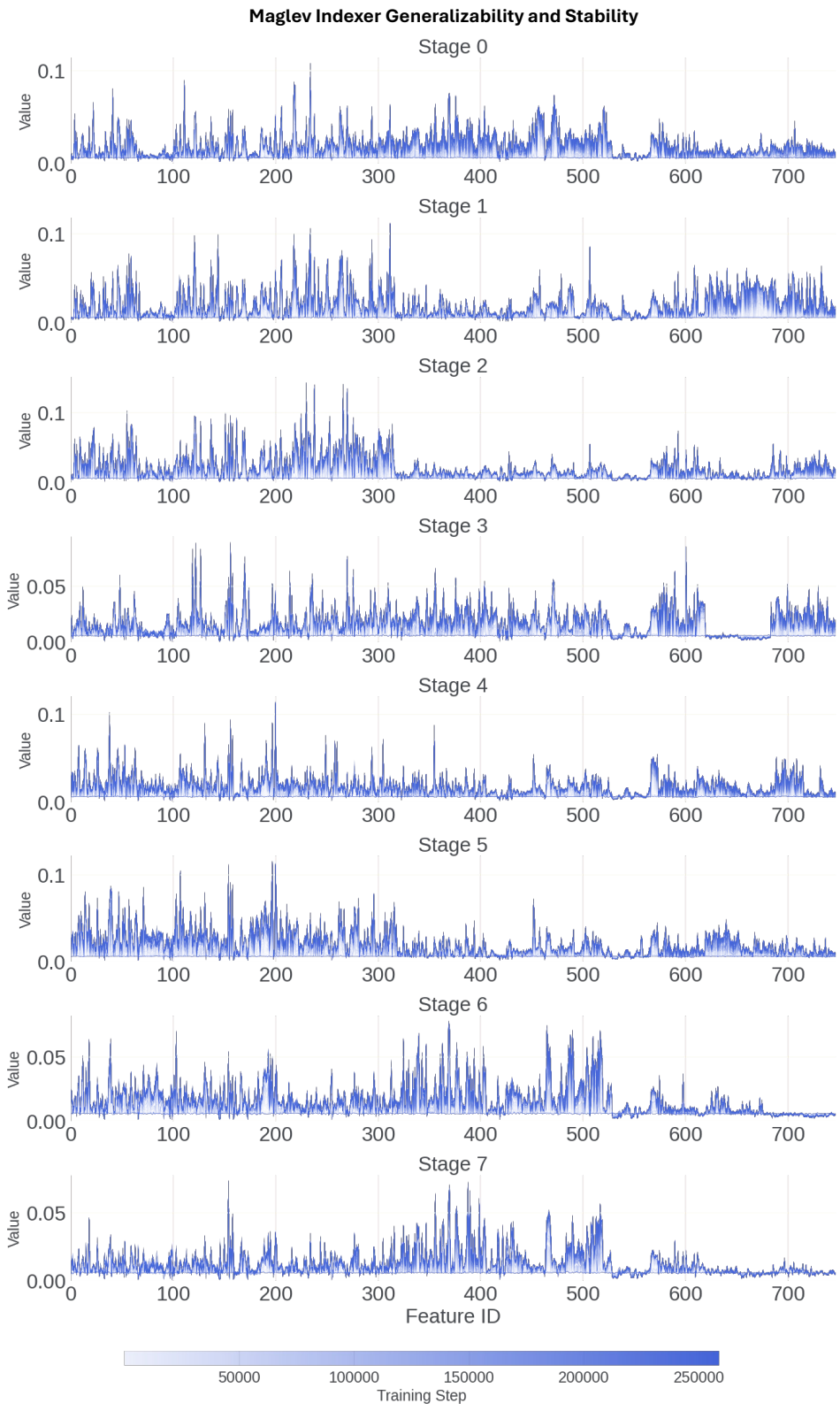

Figure 8: Affinity values stabilize early during training, yielding stable feature-to-stage assignments across eight Maglev stages.