External Large Foundation Model: How to Efficiently Serve Trillions of Parameters for Online Ads Recommendation

MINGFU LIANG, Meta, Menlo Park, CA, United States

XI LIU, Meta, Menlo Park, CA, United States

RONG JIN, Meta, Menlo Park, CA, United States

BOYANG LIU, Meta, Menlo Park, CA, United States

QIULING SUO, Meta, Menlo Park, CA, United States

QINGHAI ZHOU, Meta, Menlo Park, CA, United States

View all

Published: 23 May 2025

Citation in BibTeX format

WWW '25: The ACM Web Conference 2025
April 28 - May 2, 2025
Sydney NSW, Australia

Conference Sponsors:
SIGWEB

External Large Foundation Model: How to Efficiently Serve Trillions of Parameters for Online Ads Recommendation

Mingfu Liang*
Xi Liu*
Rong Jin
Boyang Liu
Qiuling Suo
Qinghai Zhou
Song Zhou
Meta Platforms
Menlo Park, US

Laming Chen
Hua Zheng
Zhiyuan Li
Shali Jiang
Jiyan Yang
Xiaozhen Xia
Fan Yang
Meta Platforms
Menlo Park, US

Yasmine Badr
Ellie Wen
Shuyu Xu
Hansey Chen
Zhengyu Zhang
Jade Nie
Chunzhi Yang
Meta Platforms
Menlo Park, US

Zhichen Zeng
University of Illinois
Urbana-Champaign
Champaign, US

Weilin Zhang
Xingliang Huang
Qianru Li
Shiquan Wang
Evelyn Lyu
Wenjing Lu
Rui Zhang
Meta Platforms
Menlo Park, US

Wenjun Wang
Jason Rudy
Mengyue Hang
Kai Wang
Bo Long
Wenlin Chen
Santanu Kolay
Huayu Li
Meta Platforms
Menlo Park, US

Abstract

Ads recommendation is a prominent service of online advertising systems and has been actively studied. Recent studies indicate that scaling-up and advanced design of the recommendation model can bring significant performance improvement. However, with a larger model scale, such prior studies have a significantly increasing gap from industry as they often neglect two fundamental challenges in industrial-scale applications. First, training and inference budgets are restricted for the model to be served, exceeding which may incur latency and impair user experience. Second, large-volume data arrive in a streaming mode with data distributions dynamically shifting, as new users/ads join and existing users/ads leave the system. We propose the External Large Foundation Model (ExFM) framework to address the overlooked challenges. Specifically, we develop external distillation and a data augmentation system (DAS) to control the computational cost of training/inference while maintaining high performance. We design the teacher in a way like a foundation model (FM) that can serve multiple students as vertical models (VMs) to amortize its building cost. We propose Auxiliary Head and Student Adapter to mitigate the data distribution gap between FM and VMs caused by the streaming data issue. Comprehensive experiments on internal industrial-scale applications and public datasets demonstrate significant performance gain by ExFM.

*Both authors contributed equally to the paper

CCS Concepts

• **Information systems** → **Recommender systems**.

Keywords

Ads Recommendation, Knowledge Distillation, Large-Scale Model

ACM Reference Format:
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, Bo Long, Wenlin Chen, Santanu Kolay, and Huayu Li. 2025. External Large Foundation Model: How to Efficiently Serve Trillions of Parameters for Online Ads Recommendation. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3701716.3715223

1 Introduction

Ads recommendation is an important service provided by online advertising systems, whose model performance can impact user experience. It has been actively studied to enhance model performance by advanced designs [8, 20, 32, 35, 38, 43] and scaling-up

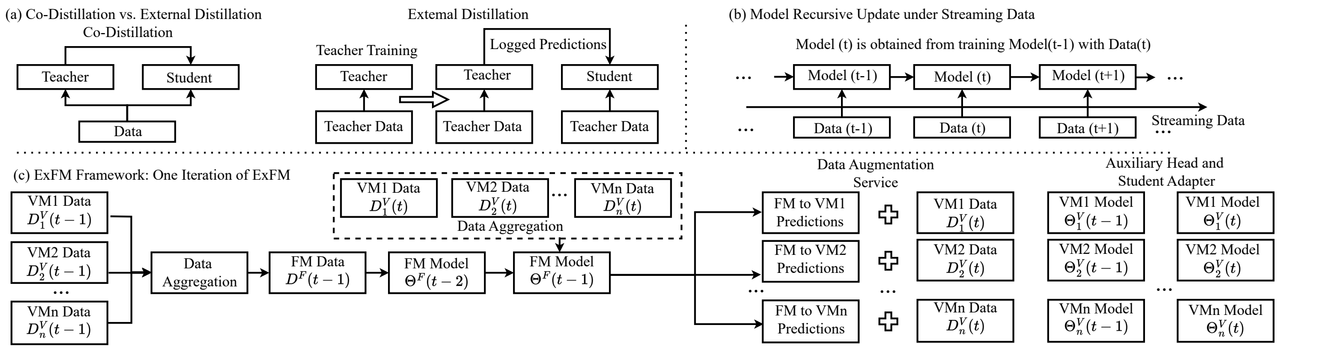

Figure 1: (a) Co-Distillation vs. External Distillation; (b) Model update under streaming data; (c) Our proposed ExFM framework that enabled trillions-parameter model serving with newly designed data augmentation system (DAS) and external distillation; The proposed auxiliary head and student adapter are developed to mitigate data distribution gap between FM and VMs.

model complexity [3, 24, 37], especially after observing the remarkable performance of trillion-parameter models such as GPT-4 [2], LLaMa [30], etc. However, prior studies often neglect two fundamental challenges in industrial-scale applications:

- C1: Restricted training and inference latency for the serving models. Industrial platforms usually have to recursively update the models, and evaluate the prediction score of O(100)~O(100K) ads for each request within a restricted latency. Exceeding latency may degrade model performance and impair user experience.
- C2: Large-volume streaming data arrive with data distributions dynamically shifting. This is largely due to the evolving nature of advertising systems: new users/ads join, and existing users/ads leave the system as time passes. This implies that multi-pass training would risk over-fitting and is the most salient difference between industry and academia.

To overcome the restricted inference latency challenge in C1, multiple prior studies employ knowledge distillation (KD) [14, 18, 21, 41]. As shown by Figure 1(a), a large teacher model is co-trained with a compact student model to be used for serving. The teacher's knowledge is continuously distilled into the student. Unfortunately, the co-training will increase the training computational cost of the serving model, which cannot meet the industrial restriction on the training latency of the serving model. Besides, ads recommendation often involves multiple serving models, each corresponding to one specific service or ranking stage (e.g., early or later/ranking stage). Building and maintaining a dedicated large teacher model for each serving model is too inefficient and non-scalable. To sum up, the ideal teacher model (1) is separately trained with the student model, i.e., external distillation, and (2) is 1-to-N, able to benefit multiple student models, like a Foundation Model (FM).

The streaming and non-stationary nature of recommendation data in C2 makes it challenging to realize the ideal teacher model. As shown by Figure 1(b), models need to be recursively trained to capture the up-to-date distribution shifting. Figure 2 illustrates an example of performance degradation under model staleness over internal datasets, where the model is trained with 74-billion examples and stops training with new examples since inference. Compared to the baseline being trained with new examples daily, its inference NE (normalized entropy) [12] loss keeps enlarging as the delay inclines. Besides, since the teacher aggregates various students' training data from different services, the distribution of the teacher model's data will be more sophisticated. Those factors lead to two potential distribution gaps between the teacher model and the student models: (1) Cross-domain Bias. For an individual student, the teacher may carry bias due to being trained by not only that student's data but an aggregation of all students' data under distribution shifting. (2) Freshness Gap. The teacher predictions used in external distillation are from a teacher with a slight delay w.r.t. training data compared to the student under training.

In light of the above limitations, we propose the External Large Foundation Model (ExFM) framework to address the challenges overlooked by prior studies. Figure 1(c) presents a big picture of the ExFM framework. To avoid additional training or inference computational costs on the serving model, ExFM employs external distillation where the teacher model is separately trained, and the teacher's predictions are offline logged as external supervision for the student training. To amortize the resources needed for training and maintenance of the teacher model, ExFM aggregates the data from multiple student traffic as the training data of the teacher, like a 1-to-N Foundation Model that can serve multiple students as the vertical models (VMs), i.e. production model. To alleviate the Cross-domain Bias transferred from FM to VMs, ExFM proposes Auxiliary Head (AH) that uses an isolated task arch to receive the FM supervision instead of direct on serving head. The simple change is found effective both provably and empirically. To mitigate the Freshness Gap between FM and VMs, ExFM proposes Student Adapter (SA) that learns to transform FM predictions before being used as VMs' supervision.

The main contributions of the paper are summarized as follows

- We re-define the ads recommendation problem to be explored under two fundamental challenges from industrial-scale applications overlooked by prior studies: (1) restricted training and inference budget for serving model, and (2) streaming data under continuously shifting distributions.
- We propose ExFM framework to overcome above challenges, where (1) the teacher is an FM to cover multiple VMs to amortize the maintaining cost; (2) external distillation (Sec. 3.2), not co-distillation, is designed with the system to avoid training or inference cost to models to be served (i.e., VM); (3) Auxiliary Head (Sec. 3.3), i.e., an isolated task instead of VM's serving head, is used to process FM' supervision, provably

alleviating the bias transfer from FM to VMs, (4) Student Adapter (Sec. 3.4) with specific learning algorithms is proposed to reduce the staleness gap between FM and VMs, with mathematical guarantee.
- We conducted extensive experiments (Sec. 4) on both internal industrial-scale datasets and public datasets and demonstrated that ExFM can enable efficient serving of a trillion-parameter model which have been deployed to Meta Platforms for more than two years. We observed promising performance enhancement by employing ExFM on VMs from different domains/tasks/stages. We also provide ablation studies to reveal the impact of hyper-parameters.

2 Problem Formulation

We re-define the ads recommendation problem to match industrial-scale applications with two overlooked fundamental challenges:

- Restricted training and inference latency for serving models. Ideally, the model enhancement should not incur any training or inference computational costs on serving models.
- Training and inference data are large volumes of streaming data with continuously shifting data distribution. Models have to keep training on fresh examples to perform well.

The new ads recommendation problem is to predict user interest in ads based on their interaction data with dynamic data distribution shift constrained by restricted training and inference latency. Formally, let $\Theta^F(t)$ be the FM parameters after training $\Theta^F(t-1)$ on FM's new data $D^F(t)$ at time t, and $\Theta_i^V(t)$ be VM i's parameters after training $\Theta_i^V(t-1)$ on VM's new data $D_i^V(t)$, $i = 1, \ldots, n$.

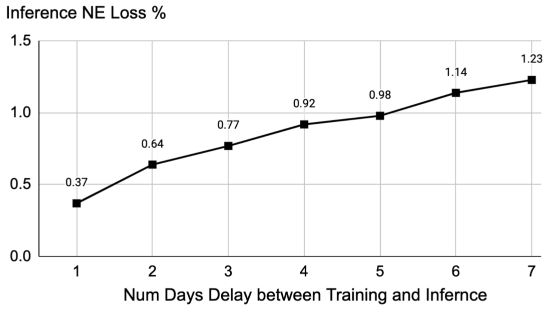

Figure 2: Click Through Rate (CTR) prediction on the internal dataset. The bigger the staleness, the larger NE Loss.

3 Methods

3.1 Overview

Figure 1(c) illustrates the details of the ExFM framework with one model iteration as an example. Specifically, FM and VMs are recursively trained on streaming data $D^F(t)$ and $\{D_i^V(t)\}_{i=1}^n$ incrementally arrived at time t. To amortize the building and maintenance costs, the teacher model is built as an FM that can make prediction on multiple VMs' traffic. Correspondingly, the training data of FM $D^F(t-1)$ is obtained by aggregating the cross-traffic VM's training data $\{D_i^V(t-1)\}_{i=1}^n$. After training FM model $\Theta^F(t-2)$ on $D^F(t-1)$, FM model parameters are updated to $\Theta^F(t-1)$.

To not incur additional training or inference costs to VMs, external distillation, as shown in Figure 1(a), is employed. Specifically, Teacher/FM is trained on $D^F(t-1)$ separately from Students/VMs. After training, Teacher/FM inferences on Students'/VMs' training data $\{D_i^V(t)\}_{i=1}^n$ to generate predictions as external supervision to Students/VMs. The external supervision, along with the VMs' training data $\{D_i^V(t)\}_{i=1}^n$ will be used to train the Students/VMs' models $\{\Theta_i^V(t-1)\}_{i=1}^n$, updating them to $\{\Theta_i^V(t)\}_{i=1}^n$. The model snapshots of Students/VMs $\{\Theta_i^V(t)\}$ are used to serve traffic for the next period $[t, t+1)$. FM's supervision logging and the merging with Students/VMs training data are achieved by Data Augmentation Service (DAS), elaborated in Section 3.2.

Since FM's training data $D^F(t-1)$ is an aggregation of VMs' data $\{D_i^V(t)\}_{i=1}^n$ from multiple traffics, for one specific traffic, FM supervision readily carries cross-domain bias. Multiple prior studies handle this problem by carefully designing the distillation losses [9]. Motivated by that, we propose a flexible solution that can cover different proposals from prior studies, called Auxiliary Head, which can provably reduce the bias transfer from FM supervision to VM and significantly enhance FM's benefits on VM in experiments. Details are provided in Section 3.3.

Also note that there is always a delay between FM training and FM inference used in VMs' training: FM $\Theta^F(t-1)$ inferences on data $\{D_i^V(t)\}_{i=1}^n$ to update VMs to $\{\Theta_i^V(t)\}_{i=1}^n$. Due to the streaming and distribution shifting nature of the industry data, this delay will create a Freshness Gap between the FM's supervision and the VMs' training and impair the beneficial impact of FM. Student Adapter is proposed to alleviate the gap and elaborated in Section 3.4.

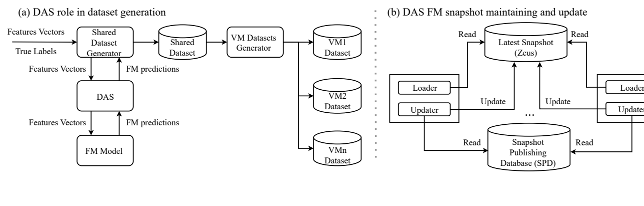

Figure 3: Data Augmentation Service (DAS)

3.2 Data Augmentation Service (DAS)

Data Augmentation Service (DAS) logs FM's supervision on the fly when preparing training data for VMs, fulfilled in two stages. As shown by Figure 3(a), first, all VMs' features and true labels are joined together into a shared dataset. There is a large time window to wait for the feedback (i.e., true label) for this training example (e.g., 5 to 90 minutes for CTR feedback, or 1 day for CVR feedback). During this time window, DAS will call FM to get FM's inference on VM's features, which usually has a large latency budget (in seconds level) sufficient to evaluate a large-scale FM. Then each VM picks its own datasets from the shared dataset for training. This ensures that the inference of FM only needs to cover a tiny percentage of ranked ads in training data, and VMs with overlapped traffic can further amortize the inference cost of the FM.

As mentioned earlier, models need to be recursively trained to capture the distribution shifting of streaming data. Consequently, FM publishes model snapshots regularly. DAS has to actively detect if there is a new snapshot and switch to it if so. In practice, it may take several minutes to switch to a snapshot, as loading also takes time. Before completeness, we want the old snapshot to still maintain high availability for FM supervision generation. To avoid wasting resources, we also want the new snapshot to be loaded only once. To this end, DAS separates the FM model snapshot publishing and the latest model snapshot database. As shown by Figure 3(b), a snapshot is regularly published in the Snapshot Publishing Database (SPD), while the latest snapshot to be loaded for generating supervision is maintained in Zeus, a distributed metadata store based on Apache Zookeeper [4] with strong consistency. Zeus maintains the snapshot with a lease to do automatic garbage collection of old snapshots. The existing lease will be extended automatically at a fixed frequency until a new snapshot is discovered. For DAS tasks, Zeus is readable and writable, while SPD is only readable. The updater of the DAS task regularly queries SPD to detect if there is a new snapshot. If a new snapshot is discovered, the updater will read it from SPD and write it into Zeus. The loader of the DAS task is responsible for loading the latest snapshot to generate supervision. DAS treats all tasks equally instead of having a special primary task to be scalable to distributed settings.

3.3 Auxiliary Head (AH)

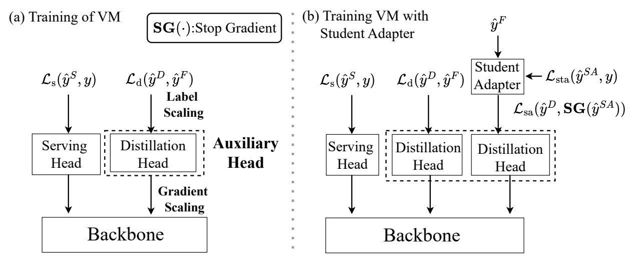
(a) Training of VM
SG(·):Stop Gradient
$\mathcal{L}_s(\hat{y}^S, y)$
$\mathcal{L}_d(\hat{y}^D, \hat{y}^F)$
Label Scaling
Serving Head
Distillation Head
Auxiliary Head
Gradient Scaling
Backbone
(b) Training VM with Student Adapter
$\hat{y}^F$
Student Adapter
$\mathcal{L}_{sta}(\hat{y}^{SA}, y)$
$\mathcal{L}_s(\hat{y}^S, y)$
$\mathcal{L}_d(\hat{y}^D, \hat{y}^F)$
$\mathcal{L}_{sa}(\hat{y}^D, \mathbf{SG}(\hat{y}^{SA}))$
Serving Head
Distillation Head
Distillation Head
Backbone

Figure 4: Auxiliary Head (AH) and Student Adapter (SA). SG means to stop the gradient from the output of SA.

The baseline of knowledge distillation is through label smoothing of the ground-truth label y and the pseudo-label $\hat{y}^F$ from FM. The model architecture of VM consists of two parts: the backbone and a serving head. We define the output vector of the backbone as $\mathbf{x}$, and $\mathbf{x}$ will be sent to the serving head. The output vector of the serving head is defined as $\hat{y}^S = \phi(\mathbf{x})$, where ϕ denotes the multi-layer perceptron (MLP). The training objective is binary cross-entropy and can be generally defined as

$$h(\hat{y}, y) = y \cdot \log(\sigma(\hat{y})) + (1 - y) \cdot \log(1 - \sigma(\hat{y})), \quad (1)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ denotes the Sigmoid activation function and $\hat{y}$ denotes the prediction. Accordingly, the training loss of VM with knowledge distillation is defined as

$$\mathcal{L}_{\text{kd}}(\hat{y}^S, \hat{y}^F, y) = h(\hat{y}^S, y) + h(\hat{y}^S, \hat{y}^F) \quad (2)$$

where $\hat{y}^S$ denotes the output of the serving head. The gradients with respect to y and $\hat{y}^F$ are entangled in the serving head and flowing back to the backbone.

Using a single head to consume both labels may induce bias from FM to VM. Therefore, we propose the Auxiliary Head (AH) that leverages a separate head to consume the pseudo-label from FM. In this case, the model architecture of VM consists of the backbone, the serving head, and a distillation head as AH. The output of the backbone $\mathbf{x}$ will be sent to different heads, and the output of AH is from the distillation head $\hat{y}^D$, where $\hat{y}^D$ is defined as $\hat{y}^D = \psi(\mathbf{x})$. ψ represents an additional MLP head which is built on top of the backbone. By disentangling the serving head and distillation loss, the serving head is only supervised by the ground-truth label y as

$$\mathcal{L}_{\text{s}}(\hat{y}^S, y) = h(\hat{y}^S, y). \quad (3)$$

The distillation head is supervised by the pseudo-label $\hat{y}^F$ as

$$\mathcal{L}_{\text{d}}(\hat{y}^D, \hat{y}^F) = h(\hat{y}^D, \hat{y}^F). \quad (4)$$

We further provide the theoretical insight that, compared to adding the auxiliary loss to the serving task for distillation, we prove that integrating the auxiliary head for knowledge distillation (KD) can alleviate the bias transfer from FM to VMs.

THEOREM 3.1. *(Informal) The VM will contain bias from the FM when the VM is trained with KD by a single serving head. In contrast, the VM is guaranteed to find the optimal solution for predicting y^{gt} with KD by auxiliary heads.*

The proof of Theorem 3.1 in the Appendix Sec. 7.3, where we prove by construction using a two-layer linear model to show that (1) the bias in the pseudo labels from the FM is partially compensated by the separated prediction head for KD, and (2) fewer "biased" gradients will flow from the prediction head for KD to the shared backbone, leading to a better-generalized performance than the single serving head for KD.

However, only isolating the supervision from $\hat{y}^F$ to y is not enough to bring effective knowledge transfer in practice. This is due to the issues originating from the intrinsic properties of the training data of Ads engagement, i.e., the majority of engagement data does not convert to actions like clicking; thus, the FM primarily predicts majority of $\hat{y}^F$ close to zero, making the KD task fitting to a difficult long-tailed distribution. These practical issues hinder the effectiveness of the KD with AH.

To remedy the above issue, we propose amplifying the FM's distillation effect by **Gradient Scaling (GS), Label Scaling (LS), and Loss Weighting (LW)**. The gradient scaling scales the gradient of the distillation head to backbone with a hyperparameter β during training to enlarge the impact of the distillation from the FM.

For label scaling, we multiply the $\hat{y}^F$ with a constant α to enlarge its magnitude and clip the $\alpha \cdot \hat{y}^F$ correspondingly to avoid the scaled label being out-of-bound, e.g., $\alpha \cdot \hat{y}^{FM}$ needs to be smaller or equal to 1. Lastly, we apply the loss weighting to $\mathcal{L}_{\text{d}}$ with a hyperparameter w. Accordingly, the final training loss $\mathcal{L}_{\text{ah}}$ for VM with AH is:

$$\mathcal{L}_{\text{ah}} = \mathcal{L}_{\text{s}}(\hat{y}^S, y) + w * \mathcal{L}_{\text{d}}(\hat{y}^D, \alpha \cdot \hat{y}^F) \quad (5)$$

3.4 Student Adapter (SA)

To further alleviate the bias from the $\hat{y}^F$, we propose to provide additional distillation using the adapted $\hat{y}^F$ based on y, called Student Adapter (SA). The output of SA is defined as $\hat{y}^{SA} = \text{MLP}(\hat{y}^F)$. As shown in Figure 4 (b), SA is trained by the loss

$$\mathcal{L}_{\text{sta}}(\hat{y}^{SA}, y) = h(\hat{y}^{SA}, y) \quad (6)$$

When training VM with SA, we first optimize $\mathcal{L}_{\text{sta}}(\hat{y}^{SA}, y)$ to update SA and get $\hat{y}^{SA}$. To use $\hat{y}^{SA}$ for training VM, we stop the gradient of $\hat{y}^{SA}$, i.e., $\text{SG}(\hat{y}^{SA})$, to avoid the gradient of VM flowing back to SA, and we consume $\text{SG}(\hat{y}^{SA})$ with another distillation head to update VM with the following loss

$$\mathcal{L}_{\text{sa}}(\hat{y}^D, \text{SG}(\hat{y}^{SA})) = h(\hat{y}^D, \text{SG}(\hat{y}^{SA})). \quad (7)$$

The algorithm is detailed in Algorithm 1.

We provide theoretical insight that SA can mitigate the freshness gap between FM and VM by considering the standard regression problem for analysis. The proof is provided in Appendix 7.4.

Theorem 3.2. *With a probability $1-\delta$, the underlying regression model w and the optimal solution w_V achieved by training VM with Student Adapter (SA) is bounded as:*

$$|w_V - w|_2 \leq O\left(\gamma\sqrt{\frac{(ds)^{1/2}}{N}\log\frac{1}{\delta}}\right),$$

where N denotes the number of training samples, d denotes the input dimension, γ is a constant, s denotes the model dimension.

Compared to the bound for the standard regression model when training with KD, the bound is reduced by the factor $(d/s)^{1/4}$, implying that if the distribution is changing on the time scale between d and $\sqrt{ds}$, the student adapter can quickly catch up the drift.

Algorithm 1 Train VM with Student Adapter for one iteration

```
Input: Pseudo-label $\hat{y}^F$ from FM, ground-truth label $y$, learnable model parameter $\Theta^{SA}$ of the Student Adapter SA, learnable parameters $\Theta^V$ of the VM.
Output: Model parameter of the Student Adapter $\Theta^{SA}$, and the model parameter of VM $\Theta^V$
1: Optimize $\mathcal{L}_{\text{sta}}(\hat{y}^{SA}, y)$ (Eqn. 6) to update $\Theta^{SA}$
2: Get SG($\hat{y}^{SA}$) by stopping gradient of the output for SA
3: Optimize $\mathcal{L}_{\text{ah}}$ (Eqn. 5) + $\mathcal{L}_{\text{sa}}$ (Eqn. 7) to update $\Theta^V$.
4: Return $\Theta^{SA}$ and $\Theta^V$
```

Table 1: [Public] Dataset Statistics

Dataset	#Samples	#Features
Amazon Electronics	3.0M	6
TaobaoAd	25.0M	22
KuaiVideo	13.7M	9

4 Experiments

We evaluate the proposed ExFM on both internal industrial-scale and public datasets to answer the following research questions:

- Q1: How effective is the proposed ExFM in elevating the performance of a single VM and multiple VMs, e.g., from different tasks, domains or stages? (Sec. 4.2)
- Q2: How effective are the proposed Auxiliary Head and Student Adapter? (Sec. 4.3)
- Q3: How do the values of hyperparameters impact the ExFM performance? (Sec. 4.4)

For better readability, each part of discussion will first cover internal datasets, then public datasets. The caption of each figure and table starts from [Internal] if obtained on internal datasets and [Public] on public datasets.

4.1 Experiment Setup

4.1.1 ***Datasets.*** We conduct comprehensive experiments on internal industrial-scale datasets with billions of training examples, from ads CTR prediction tasks. Each training example contains O(1k) features for a user and ad pair, and various types of feedback such as click, post-click conversion, etc. Training examples of individual VMs come from the corresponding services or stages and may contain service-dedicated features and feedback. Training examples of the FM aggregate the training examples of VMs. Billions of training examples arrive daily for both VMs and FM. All training is one-pass, and training batches are ordered by arrival time.

For public datasets, we use TaobaoAd [29], Amazon Electronics [11] and KuaiVideo [1] as they are widely adopted and have the timestamp feature that facilitates our experiments for streaming setting. The dataset statistics are briefly summarized in Table 1. For the streaming setting, we split each dataset into 8 splits (days) and divide them into the FM train set, VM train set, and VM test set based on the UNIX timestamps. For example, FM is trained on the first 4 days' data and inference on 5th day's data to generate supervision. VM is trained on the 5th day with FM's supervision and inference on the 6th day to get the AUC/LogsLoss reading. We can repeat this process for 3 times until Day 8. On TaobaoAd and Amazon Electronics, we train FM and VMs on the CTR prediction task. To validate FM's impact on different-domain VMs, we split TaobaoAd dataset by the value of the categorical feature 'new_user_class_level'. The original 'new_user_class_level' feature has five levels; we combine them into three levels as domains to ensure that each domain has almost the same training data size. To experiment the FM's impact on different-task VMs, we use KuaiVideo [1] due to its various feedbacks. We use prediction of 'Click,' 'Follow,' and 'Like,' as three different tasks, each of which has a dedicated VM to handle.

4.1.2 ***Metrics.*** We adopt three widely-used evaluation metrics:

- `NE` (normalized entropy) [12], is the `LogLoss` in Eqn. 1 normalized by the entropy of the average empirical CTR of the training dataset, providing a data-insensitive evaluation. Lower the better, and an NE gain of 0.02% is considered significant for experiments on internal datasets.
- `AUC` is an aggregate measure of model performance in correctly classifying positive and negative samples across all thresholds, used for public datasets. Higher the better.
- `LogLoss` (cross-entropy loss) measures the distance between model prediction $\hat{y}$ and actual label y by Eqn. 1, which will be used for public datasets. Lower the better.

4.1.3 ***Model Arch.*** ExFM is model-agnostic and our purpose is to show the delta impact of FM on VM. In our problem setting, FM is assumed to have larger capacity and demand more computation resource for training and inference. To be consistent, FM we experimented with on internal datasets is often 15 to 600 times larger than the VM. FM enjoys more advanced and resource intensive model architecture such as an internal version of Interformer [36], SUM[39], Wukong [37], DHEN [38], etc., while VM adopts a shrunk version of FM or less resource intensive structure such as DLRM [22].

On public datasets, we use the models that attain SOTA performance on TaobaoAd, Amazon Electronics, and KuaiVideos as FMs based on BARS [40] benchmark. Specifically, we use DMIN [34], DCN [31], and DCNv2 [32] as different variants of FM. Regarding VMs, Factorization Machine (FaM) [25] and its two variants, i.e., FmFaM [27] and DeepFaM [10] are considered, as they are less complex and low latency to SOTA. For instance, the parameter of

FaM is 4.23M on Amazon Electronics while DMIN is 5.94M, with a near 30% parameter decrease. Our purpose is to study the delta impact of FM on VMs under different choices of the two. More details of the models and training settings are in Appendix 7.2.

4.2 Effectiveness of FM on VMs

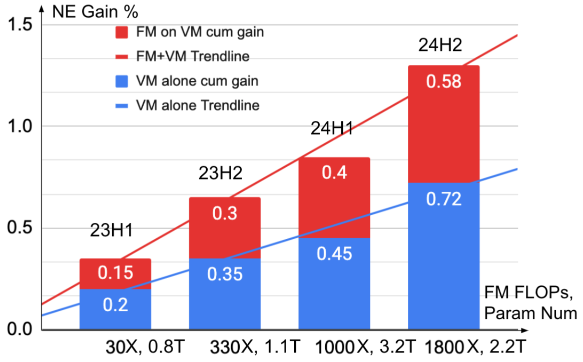

Figure 5: [Internal] Inference NE gain of different-size FM on VM iterations ($1X = 60M$ training FLOPs, 1T is 1 Trillion)

4.2.1 ***One FM for One VM.*** On internal datasets, we evaluated the impact of different-sized FMs on elevating VM performance. As shown by Figure 5, FM model size ranges from 30X to 1800X where $1X = 60M$ training FLOPs. FM model parameter number ranges from 0.8T to 3.2T where 1T is 1 Trillion. FM scale-up is employing advanced model arches with more features and larger embedding dimensions. VMs are all 30X and from past halves' real iterations, e.g., 23H1 implies the iteration of the first half in 2023. VMs are iterated by refreshing features and adopting micro model arch changes. The baseline of both FMs and VMs is the VM at 22H2. NE Gain % refers to the percentage of NE improvement attained vs. baseline. "VM alone" NE Gain % means the NE improvement of VM obtained without FM. Cumulative gain means the number includes the gain since 23H1. Both FM and VMs are trained with $> 300B$ streaming data and inference on the next-day's new arriving data. Although both VMs and FM are changing from half to half, the value of FM on VM gain is obtained by apple-to-apple comparison. We observe that (1) FMs can generate consistent NE gains on VMs from half to half, (2) compared to the trend of the world without FM (blue trendline), the one with FM has steeper slope (red trendline), implying the role of FM on bending the trending curve.

On public datasets, we evaluated the impact of FM on VMs under different FM-VM pairs in Table 2. Data is described in Sec. 4.1.1 and VM is trained on the 5th day with FM's supervision and inference on the 6th day to get the AUC/LogsLoss reading. Note that the original performance of VMs without FM is limited, because the VMs' model is chosen not to have any SOTA and complex model architectures for the consistency with the problem settings. In the table, "distill w/ $\mathcal{L}_{\text{ah}} + \mathcal{L}_{\text{sa}}$" means w/ FM, and "w/o distill" means w/o FM. From the table, we observe consistent AUC and Logloss improvement from having FM across different FM-VM choices, demonstrating that the effectiveness of ExFM can hold in general.

4.2.2 ***One FM for N VMs.*** We evaluate the impact of one FM on multiple VMs across multi-stages in the ads system, including Retrieval, Early, and Later stages, each of which reduces ads candidates, from O(100k) to O(1k), from O(1k) to O(100), and from O(100) to O(10), respectively. Specicially, we use the 1000X, 3.2T FM in Figure 5 and VMs from three stages for experiments on internal dataset. Retrieval VM and Early stage VM employ the Two-Tower Sparse Network (TTSN) [33], and Later stage VM uses DLRM [22] as the corresponding VM in Figure 5. Training and inference data preparation also follow those of Figure 5. In Figure 6, we observe that the FM brings 0.11% to 0.25% additional NE gain to VMs of different stages. The earlier a stage is, the more gains the FM generates.

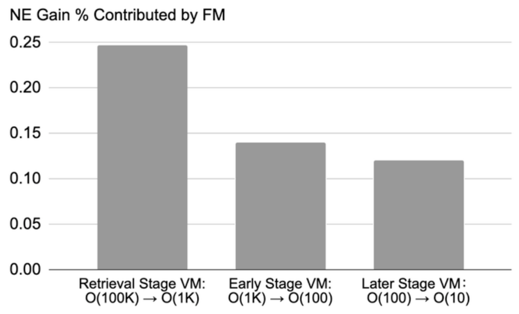

Figure 6: [Internal] Inference NE gain of 1000X, 3.2T FM on cross-stage VMs.

To verify the benefits of FM to multiple VMs from different domains, we split the TaobaoAd data into different domains as described in Sec. 4.1.1. To verify the same thing for different tasks, we use the KuaiVideo dataset where multiple types of feedback are available for prediction. We use the hard-parameter shared DMIN [23] as the FM and FaM [25] dedicatedly trained from scratch for each domain or task as the VM. Table 3 and Table 4 illustrate the results for different domains and tasks separately. In both tables, we observe that w/ FM outperforms the w/o FM in all tasks and domains, demonstrating the effectiveness of FM for multiple VMs.

4.3 Effectiveness of of AH and SA

To understand the role of AH in the success of ExFM, on internal datasets, we compare the VM's NE after including AH (Eqn. 5) vs. w/o AH (Eqn. 2) under the same 1000X, 3.2T FM and corresponding VM from Figure 5. Figure 7 provides the comparison, where the x-axis is the number of streaming training examples and the value of NE Change at each point is obtained by using the current snapshot to inference on next-batch data, thus equivalent to the inference NE, but in a streaming mode. Since NE is the lower the better, the negative 'NE Change %' (-4%) implies that AH brings a large and stable performance gain compared to baseline. Similarly, to understand the role of SA, we compare the VM's NE after including SA vs. w/o SA under the same 1800X, 2.2T FM, and corresponding VM from Figure 5. Figure 8 shows the comparison, and we observe that with SA, the NE performance of VM on streaming data improves by 0.08%, with an enlarging trend, which is considered significant.

To reproduce the results on public datasets, we split the dataset as discussed in Sec. 4.1.1 by timestamps to simulate the streaming setting. We apply AH and SA over different pairs of public models on public datasets as described in Section 4.1.3, and the performance of VMs is reported in Table 2. Specifically

- distill with $\mathcal{L}_{\text{kd}}$ (Eqn. 2), i.e., the distillation loss
- distill with $\mathcal{L}_{\text{ah}}$ (Eqn. 5), i.e., the AH
- distill with $\mathcal{L}_{\text{ah}} + \mathcal{L}_{\text{sa}}$ (Eqn. 7), i.e., AH and SA.

Table 2: [Public] ExFM performance under different FM/VM, AH/SA and datasets choices.

	Dataset ⟶	TaobaoAd						Amazon Electronics					
	Vertical Models ⟶	FaM [25]		FmFaM [27]		DeepFaM [10]		FaM [25]		FmFaM [27]		DeepFaM [10]	
Teacher Model	Methods	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss
DMIN [34]	w/o distill	0.5389	0.7228	0.5477	0.5388	0.5488	0.5713	0.8331	0.5493	0.8491	0.5311	0.8502	0.5303
	distill w/ $\mathcal{L}_{kd}$	0.5408	0.6484	0.5482	0.5349	0.5576	0.2943	0.8349	0.5473	0.8501	0.5305	0.8519	0.5283
	distill w/ $\mathcal{L}_{ah}$	0.5449	0.6451	0.5494	0.5223	0.5630	0.2885	0.8399	0.5431	0.8538	0.5232	0.8551	0.5165
	distill w/ $\mathcal{L}_{ah}$ + $\mathcal{L}_{sa}$	**0.5462**	**0.6411**	**0.5521**	**0.5208**	**0.5648**	**0.2871**	**0.8418**	**0.5379**	**0.8557**	**0.5141**	**0.8573**	**0.5119**
DCN [31]	w/o distill	0.5389	0.7228	0.5477	0.5388	0.5488	0.5713	0.8331	0.5493	0.8491	0.5311	0.8502	0.5303
	distill w/ $\mathcal{L}_{kd}$	0.5397	0.6587	0.5479	0.5359	0.5556	0.3183	0.8339	0.5486	0.8497	0.5288	0.8514	0.5290
	distill w/ $\mathcal{L}_{ah}$	0.5438	0.6565	0.5495	0.5228	0.5612	0.2918	0.8374	0.5451	0.8534	0.5239	0.8548	0.5181
	distill w/ $\mathcal{L}_{ah}$ + $\mathcal{L}_{sa}$	**0.5458**	**0.6449**	**0.5517**	**0.5213**	**0.5624**	**0.2901**	**0.8401**	**0.5390**	**0.8551**	**0.5153**	**0.8570**	**0.5115**
DCNv2 [32]	w/o distill	0.5389	0.7228	0.5477	0.5388	0.5488	0.5713	0.8331	0.5493	0.8491	0.5311	0.8502	0.5303
	distill w/ $\mathcal{L}_{kd}$	0.5401	0.6580	0.5481	0.5351	0.5561	0.3119	0.8343	0.5480	0.8503	0.5298	0.8522	0.5272
	distill w/ $\mathcal{L}_{ah}$	0.5440	0.6538	0.5498	0.5219	0.5626	0.2899	0.8392	0.5441	0.8541	0.5225	0.8568	0.5131
	distill w/ $\mathcal{L}_{ah}$ + $\mathcal{L}_{sa}$	**0.5463**	**0.6401**	**0.5525**	**0.5199**	**0.5639**	**0.2879**	**0.8411**	**0.5410**	**0.8563**	**0.5139**	**0.8582**	**0.5104**

Table 3: [Public] Impact of FM to cross-domain VMs

Methods	Domain	AUC	logloss
w/o FM	1	0.5025	2.2831
w/ FM	1	**0.5416**	**1.8964**
w/o FM	2	0.5024	2.5663
w/ FM	2	**0.5306**	**1.8812**
w/o FM	3	0.4923	3.6295
w/ FM	3	**0.5456**	**1.9003**

Table 4: [Public] Impact of FM to cross-task VMs

Methods	Task	AUC	Logloss
w/o FM	Click	0.7034	0.4693
w/ FM	Click	**0.7101**	**0.4678**
w/o FM	Follow	0.7613	0.00891
w/ FM	Follow	**0.7743**	**0.00885**
w/o FM	Like	0.8531	0.01871
w/ FM	Like	**0.8647**	**0.01869**

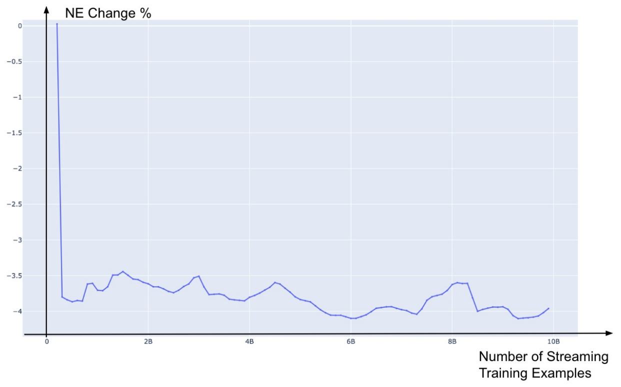

Figure 7: [Internal] NE change of AH with 1000X 3.2T FM

We observe that the AH and SA are consistently superior to the baseline with $\mathcal{L}_{kd}$ among different pairs of FMs and VMs on various datasets, demonstrating their strong effectiveness in enhancing FM benefits on VMs. As the breakdown, it is found that (1) the vanilla distillation w/o AH or SA has very small benefits, e.g., 0.35% AUC

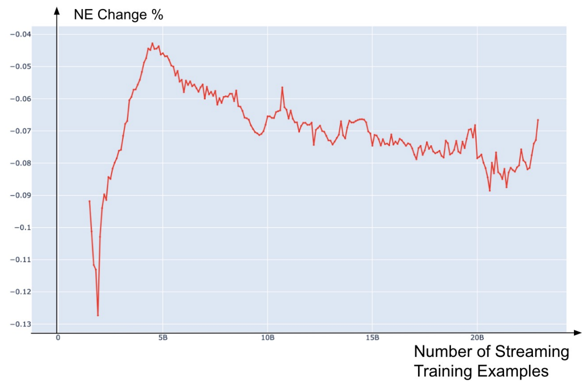

Figure 8: [Internal] NE change of SA with 1800X, 2.2T FM

gain for DMIN as FM and FaM as VM on TaobaoAd, (2) the major leap is by AH, e.g., boosting the AUC gain from 0.35% to 1.11%, and (3) SA further brings significant gains on top of AH, e.g., lifting the AUC gan from 1.11% to 1.35%.

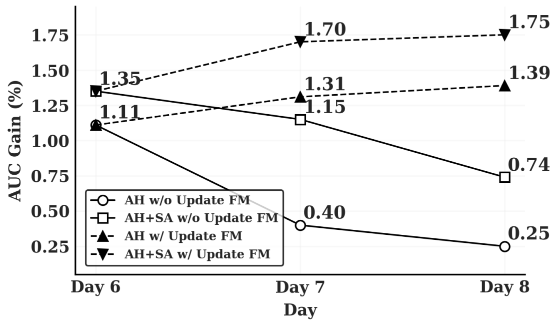

Figure 9: [Public] Impact of SA under FM staleness.

The data points in Table 2 are obtained by one-time inference on the 6th day's data. We are curious how is the performance of SA with a bigger freshness gap between FM and VM. To study this, we use DMIN as FM and FaM as VM, stop the FM from being trained on the new day's data, and compare their performance w/ and w/o SA in Figure 9. We observe that (1) the benefits of FM on VM diminish as the time that FM stops training on new data is longer, (2) SA can generate additional gain no matter whether FM updates or not, but (3) SA cannot revert the diminishing trend when the FM is not updated with new day's data.

4.4 Impact of hyper-parameters

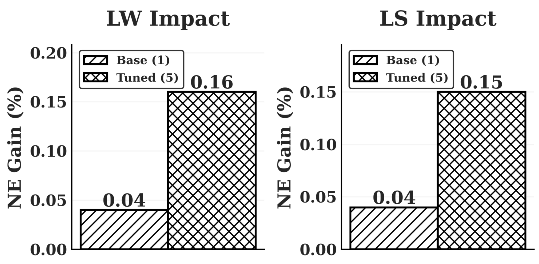

Figure 10: [Internal] Impact of LW and LS on VM's NE

We notice that the benefits of FM on VM may be sensitive to the choice of GS, LS, and LW as defined in Eqn. 5. On internal datasets, Figure 10 demonstrates the positive impact of increasing LW and LS. We observed that the benefits of FM on VM are improved correspondingly when we increase the value of LW and LS with others fixed. Figure 11 shows the impact of GS, where there seems to be a sweet spot. The NE gain first increases significantly when GS increases and then tends to plateau when GS is large enough.

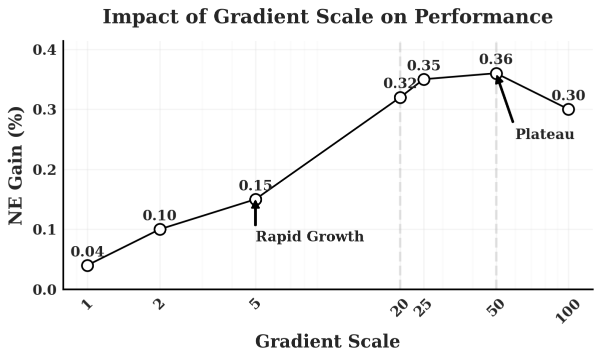

Figure 11: [Internal] Impact of GS on VM's NE

On the internal dataset, the test for the combination effect of LW, LS, and GS is often restricted by the limited training resources. We explore that on the public dataset. Figure 12 illustrates our experiments on TaobaoAd: (1) The simple combination, i.e., ($LS = 1, GS = 1, LW = 1$), can only achieve moderate improvement on the VM. (2) Aggressive scaling, e.g., ($LS = 10, GS = 10, LW = 10$) or ($LS = 1, GS = 1, LW = 100$), will even be detrimental to the VM's performance. Both observations imply that an appropriate combination of these three scaling techniques could yield significant accuracy improvement and it is important to identify a proper combination for practical usage.

5 Related Works

Due to the paper limit, we include the review of knowledge distillation for industrial recommendation system in the Appendix 7.1. Among existing studies, ours are closest to [19] and [17]. Compared to [19], our study has the following salient differences: (1) Different settings for the teacher model. To amortize building and maintaining resources, the teacher model in ExFM is an FM that uses an aggregation of student models' feature sets. As a result, the teacher itself often has feature gap and dimension mismatch when compared to an individual VM, so ExFM does not generate

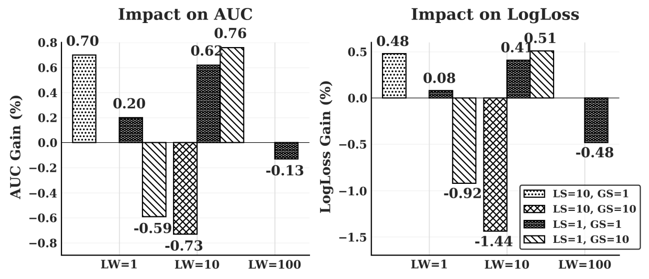

Figure 12: [Public] Joint impact of different factors, i.e., LW, LS, and GS on VM performance.

student models from the teacher model like [19]. (2) Teacher update does not depend on students. [19] instead couples the teacher model's update with students' by replay learning. Such dependency will increase overhead of teacher update and bring in staleness, as illustrated by Figure 2, implying a risk of huge performance loss. (3) Consideration of data distribution shifting in an industrial streaming setting. We develop Student Adapter to handle that and provide theoretical guarantees. Compared to [17], our differences and contributions are as follows. Its auxiliary distillation looks similar to our AH, but our AH is an isolated task arch that consumes supervision from FM in a dedicated fashion, not simultaneously consuming true labels like in [17]. [17] lacks theoretical proof to justify its proposal and does not provide a solution to mitigate the distribution gap between FM and VMs due to data distribution shifting. Instead, we develop and prove Student Adapter can achieve the goal. Moreover, [17] does not contain sufficient details and lacks comprehensive benchmark experiments on external datasets.

6 Conclusion

In this paper, we propose the ExFM framework to address two fundamental but overlooked challenges by prior studies on scaling-up Ads recommendation models – (C1) restricted training and inference budget and (C2) streaming data with distribution shifting. To overcome C1, ExFM employs the external distillation and the data augmentation service, where teacher training separates from student training and one teacher can supervise multiple VMs, like an FM. To alleviate the distribution gap between FM and VMs caused by C2, ExFM proposes Auxiliary Head (AH) and Student Adapter (SA), with mathematical guarantees on benefits provided. ExFM, including its core techniques such as AH and SA, achieved outstanding performance on industrial-scale datasets over multiple tasks and stages. It enabled the serving of a trillion-parameter model without increasing serving latency or negatively impacting user experience, and established the capability to serve LLM-scale ads model in the future. We also experimented on public datasets and were able to reproduce ExFM's outstanding performance, further demonstrating the effectiveness of the proposed framework.

References

[1] [n. d.]. Kuaishou, howpublished = https://www.kuaishou.com/activity/uimc.
[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
[3] Rohan Anil, Sandra Gadanho, Da Huang, Nijith Jacob, Zhuoshu Li, Dong Lin, Todd Phillips, Cristina Pop, Kevin Regan, Gil I Shamir, et al. 2022. On the factory

floor: ML engineering for industrial-scale ads recommendation models. *arXiv preprint arXiv:2209.05310* (2022).
[4] Apache. 2023. ZooKeeper. https://zookeeper.apache.org/
[5] Peter Bartlett, Dave Helmbold, and Philip Long. 2018. Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. In *ICML*. PMLR, 521–530.
[6] Gang Chen, Jiawei Chen, Fuli Feng, Sheng Zhou, and Xiangnan He. 2023. Unbiased knowledge distillation for recommendation. In *WSDM*. 976–984.
[7] Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. 2024. Scaling laws for dense retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1339–1349.
[8] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*. 299–315.
[9] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision* 129, 6 (2021), 1789–1819.
[10] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 1725–1731.
[11] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.
[12] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. 2014. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*. 1–9.
[13] Geoffrey Hinton. 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531* (2015).
[14] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2021. Item-side ranking regularized distillation for recommender system. *Information Sciences* 580 (2021), 15–34.
[15] SeongKu Kang, Wonbin Kweon, Dongha Lee, Jianxun Lian, Xing Xie, and Hwanjo Yu. 2023. Distillation from heterogeneous models for top-k recommendation. In *Proceedings of the ACM Web Conference 2023*. 801–811.
[16] SeongKu Kang, Wonbin Kweon, Dongha Lee, Jianxun Lian, Xing Xie, and Hwanjo Yu. 2024. Unbiased, Effective, and Efficient Distillation from Heterogeneous Models for Recommender Systems. *ACM Transactions on Recommender Systems* (2024).
[17] Nikhil Khani, Li Wei, Aniruddh Nath, Shawn Andrews, Shuo Yang, Yang Liu, Pendo Abbo, Maciej Kula, Jarrod Kahn, Zhe Zhao, et al. 2024. Bridging the Gap: Unpacking the Hidden Challenges in Knowledge Distillation for Online Ranking Systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 758–761.
[18] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2021. Bidirectional distillation for top-K recommender system. In *Proceedings of the Web Conference 2021*. 3861–3871.
[19] Gyuseok Lee, SeongKu Kang, Wonbin Kweon, and Hwanjo Yu. 2024. Continual Collaborative Distillation for Recommender System. In *SIGKDD*. 1495–1505.
[20] Zeyu Li, Wei Cheng, Yang Chen, Haifeng Chen, and Wei Wang. [n. d.]. Interpretable click-through rate prediction through hierarchical attention.
[21] Congcong Liu, Yuejiang Li, Jian Zhu, Fei Teng, Xiwei Zhao, Changping Peng, Zhangang Lin, and Jingping Shao. 2022. Position awareness modeling with knowledge distillation for CTR prediction. In *Proceedings of the 16th ACM Conference on Recommender Systems*. 562–566.
[22] 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. 2019. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091* (2019).
[23] Aviv Navon. 2024. *Parameter Sharing in Deep Learning*. https://avivnavon.github.io/blog/parameter-sharing-in-deep-learning/ Accessed: 2024-11-18.
[24] Junwei Pan, Wei Xue, Ximei Wang, Haibin Yu, Xun Liu, Shijie Quan, Xueming Qiu, Dapeng Liu, Lei Xiao, and Jie Jiang. 2024. Ads recommendation in a collapsed and entangled world. In *SIGKDD*. 5566–5577.
[25] Steffen Rendle. 2010. Factorization machines. In *2010 IEEE International conference on data mining*. IEEE, 995–1000.
[26] Kyuyong Shin, Hanock Kwak, Su Young Kim, Max Nihlén Ramström, Jisu Jeong, Jung-Woo Ha, and Kyung-Min Kim. 2023. Scaling law for recommendation models: Towards general-purpose user representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 4596–4604.
[27] Yang Sun, Junwei Pan, Alex Zhang, and Aaron Flores. 2021. FM2: Field-matrixed factorization machines for recommender systems. In *Proceedings of the web conference 2021*. 2828–2837.
[28] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *SIGKDD*. 2289–2298.
[29] Tianchi. [n. d.]. Ad display/click data on taobao.com, 2018. https://tianchi.aliyun.com/dataset/56.
[30] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
[31] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*. 1–7.
[32] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. 2021. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*. 1785–1797.
[33] Ruoxi Wang, Zhe Zhao, Xinyang Yi, Ji Yang, Derek Zhiyuan Cheng, Lichan Hong, Steve Tjoa, Jieqi Kang, Evan Ettinger, and H Chi. 2019. Improving Relevance Prediction with Transfer Learning in Large-scale Retrieval Systems. In *Proceedings of the 1st Adaptive & Multitask Learning Workshop*.
[34] Zhibo Xiao, Luwei Yang, Wen Jiang, Yi Wei, Yi Hu, and Hao Wang. 2020. Deep multi-interest network for click-through rate prediction. In *CIKM*. 2265–2268.
[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 974–983.
[36] Zhichen Zeng, Xiaolong Liu, et al. 2024. InterFormer: Towards Effective Heterogeneous Interaction Learning for Click-Through Rate Prediction. *arXiv preprint arXiv:2411.09852* (2024).
[37] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Shen Li, Yanli Zhao, Yuchen Hao, Yantao Yao, Ellie Dingqiao Wen, Jongsoo Park, Maxim Naumov, and Wenlin Chen. 2024. Wukong: Towards a Scaling Law for Large-Scale Recommendation. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=8iUgr2nuwo
[38] Buyun Zhang, Liang Luo, Xi Liu, Jay Li, Zeliang Chen, Weilin Zhang, Xiaohan Wei, Yuchen Hao, Michael Tsang, Wenjun Wang, et al. 2022. DHEN: A deep and hierarchical ensemble network for large-scale click-through rate prediction. *arXiv preprint arXiv:2203.11014* (2022).
[39] Wei Zhang, Dai Li, Chen Liang, Fang Zhou, Zhongke Zhang, Xuewei Wang, Ru Li, Yi Zhou, Yaning Huang, Dong Liang, et al. 2024. Scaling User Modeling: Large-scale Online User Representations for Ads Personalization in Meta. In *Companion Proceedings of the ACM on Web Conference 2024*. 47–55.
[40] Jieming Zhu, Quanyu Dai, Liangcai Su, Rong Ma, Jinyang Liu, Guohao Cai, Xi Xiao, and Rui Zhang. 2022. BARS: Towards Open Benchmarking for Recommender Systems, Enrique Amigó, Pablo Castells, Julio Gonzalo, Ben Carterette, J. Shane Culpepper, and Gabriella Kazai (Eds.). ACM, 2912–2923. doi:10.1145/3477495.3531723
[41] Jieming Zhu, Jinyang Liu, Weiqi Li, Jincai Lai, Xiuqiang He, Liang Chen, and Zibin Zheng. 2020. Ensembled CTR prediction via knowledge distillation. In *CIKM*. 2941–2958.
[42] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 2759–2769. doi:10.1145/3459637.3482486
[43] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*. 3162–3172.

7 Appendix

7.1 More Related Works

Inspired by the success of large-scale model in NLP [2, 30], more and more recent studies have focused on scaling-up recommendation model capacity for better performance [3, 7, 24, 26, 37, 39]. One fundamental challenge to employ them in industrial-scale applications is the restricted training and inference budget for serving models. To overcome that, multiple prior studies explored the adaption of knowledge distillation (KD) [13] in recommendation problems, where a large teacher model continuously supervises a compact student model by co-training and finally only the student model is used for serving [6, 14–16, 18, 21, 28, 41]. One limitation of those studies is that they primarily focused on static environments, while in industrial applications, large-volume data continuously arrive. [19] proposed to handle that by a continual learning framework.

Another limitation is that those studies follow co-training based distillation. It will increase the training cost of the serving model, as well as the risk of model staleness due to iteration delay, implying performance loss when new incoming data has distribution shifting. [17] proposed to handle that by external distillation where teacher training happens separately from supervising students.

7.2 Details of models and training configurations

We use the BARS benchmark [40] and the FuxiCTR [42] to implement all the public models. We follow the default training and model configurations (e.g., learning rate, training epoch, etc.) in BARS for each public dataset and model used for experiments.

7.3 Proof of Theorem 3.1

We consider a linear model to prove Theorem 3.1 by construction. Let $x \in \mathbb{R}^D$, where D denotes the input dimension. x will be first linearly mapped to a vector of $d \ll D$ dimension by a matrix $Z \in \mathbb{R}^{d\times D}$. We assume that the ground-truth label y is given by $y = u_1^\top Zx$, where $u_1 \in \mathbb{R}^d$. For convenience of discussion, let's assume that we have $d-1$ teacher models: for an input x, they output $d-1$ soft labels as $w_k^\top Zx, k = 2, \dots, d$. For simplicity, we assume that $w_k = u_1 + \mu u_k, k \in [d]$, with $\mu > 0$ being a small weight and $u_1, \dots, u_d$ form an orthonormal basis for the space of $\mathbb{R}^d$.

Proof. Let $(\alpha_1, \dots, \alpha_d) \in \Delta_d$ be the weights to linearly combine the ground-truth labels with teacher labels, where Δ is a simplex in $\mathbb{R}^d$. For a given input x, its smoothed label is $\widetilde{y}(x) = \left(u_1 + \mu\sum_{k=2}^d \alpha_k u_k\right)^\top Zx$. We fit $\widetilde{y}(x)$ by a two-layer linear model, i.e., $\min_{v\in\mathbb{R}^d, H\in\mathbb{R}^{d\times D}} \frac{1}{T}\sum_{t=1}^T \left|v^\top Hx_t - \widetilde{y}(x_t)\right|^2$, where T denotes the number of training samples. According to the theory of deep linear model [5], the resulting model, trained by gradient descent, will be able to find a solution with zero loss, i.e., $H^\top v = Z^\top\left(u_1 + \mu\sum_{k=2}^d \alpha_k u_k\right)$, if $\mathrm{E}\left[xx^\top\right]$ is non-singular, and we have enough number of training examples. The linear prediction model $H^\top v$ is different from $Z^\top u_1$, and the difference between the two linear prediction models arises from the bias introduced by the label-smoothing.

Now we denote $w_t^1, t = 1, \dots, T$ the solution for the prediction head for the ground-truth labels evolving over time (i.e., each training sample), and by $w_t^k, k = 2, \dots$ the dynamic solutions for the $d-1$ teacher labels. By the gradient descent method, we have

$$\begin{aligned} Z_{t+1} &= Z_t - 2\eta\sum_{k=1}^d \left(\left\langle w_t^k, Z_t x_t\right\rangle - \langle w_k, Zx_t\rangle\right) w_t^k x_t^\top \\ &= \left(I - 2\eta\sum_{k=1}^d w_t^k\left[w_t^k\right]^\top\right) Z_t x_t x_t^\top + 2\eta\sum_{k=1}^d w_t^k w_k^\top Z x_t x_t^\top. \end{aligned} \quad (8)$$

By assuming $x_t \sim \mathcal{N}(0, I)$, $\mathrm{E}_t\left[Z_{t+1}\right] = \left(I - 2\eta\sum_{k=1}^d w_t^k\left[w_t^k\right]^\top\right) Z_t + 2\eta\sum_{k=1}^d w_t^k w_k^\top Z$. As w_t^k is updated much more frequently than Z_t, we have $w_t^k \approx \left(Z_t Z_t^\top\right)^{-1} Z_t Z^\top w_k = w_k + \left(Z_t Z_t^\top\right)^{-1} Z_t\left(Z - Z_t\right)^\top w_k$, implying that when Z_t is close to Z, we will have all w_t^k being close to w_t. As a result, we have $\mathrm{E}_t\left[Z_{t+1}\right] \approx \left(I - 2\eta\sum_{k=1}^d w_k\left[w_k\right]^\top\right) Z_t + 2\eta\sum_{k=1}^d w_k w_k^\top Z = (1 - 2\eta A)Z_t + 2\eta AZ$, where $A = \sum_{k=1}^d w_k w_k^\top$ is a full rank matrix, indicating that Z_t will converge to Z and thus all w_t^k will converge to w_k. This result indicates that by using auxiliary heads, we will guarantee to find the optimal solution w_1 and Z for predicting the ground-truth labels. □

7.4 Proof of Theorem 3.2

We consider the standard regression problem for analysis. Let $\left\{x_i \in \mathbb{R}^d, i = 1, \dots, N\right\}$ be the input sampled from a normal distribution $\mathcal{N}(0, I)$. Let the observed output $y_i = x_i^\top\left(w + z_i\right)$, where w is the underlying regression model and $z_i \sim \mathcal{N}\left(0, \gamma^2 I\right)$ is an independently sampled vector from a normal distribution that leads to the noise in the observed output value y_i. In this study, we assume $\gamma \gg 1$. If we learn a regression model directly from the training data, the resulting solution w_1 is given by $w_1 = \left(\sum_{i=1}^N x_i x_i^\top\right)^{-1}\sum_{i=1}^N x_i y_i = w + \left(\sum_{i=1}^N x_i x_i^\top\right)^{-1}\sum_{i=1}^N x_i x_i^\top z_i$. Based on the Matrix Chernoff bound, with probability $1-\delta$, we have $\left|\frac{1}{N}\sum_{i=1}^N x_i x_i^\top - I\right|_2 \le O\left(\sqrt{\frac{d}{N}\log\frac{1}{\delta}}\right)$, and $\left|\sum_{i=1}^N x_i x_i^\top z_i\right|_2 \le O\left(\gamma\sqrt{\frac{d}{N}\log\frac{1}{\delta}}\right)$. When $N > \Omega(d\log(1/\delta))$, we have $|w_1 - w| \le O\left(\gamma\sqrt{\frac{d}{N}\log\frac{1}{\delta}}\right)$, implying that to achieve a recovery error ϵ, we need at least $\Omega\left(\frac{\gamma^2 d}{\epsilon^2}\log\frac{1}{\delta}\right)$ number of training samples. Now we provide the complete proof.

Proof. As the teacher model will be equipped with the student adapter, i.e., an MLP, to adapt its prediction, we assume the teacher model is parameterized as $\widehat{y} = \widehat{w}^\top x + u^\top Hx$, where $H \in \mathbb{R}^{s\times d}$ is a given orthonormal matrix (i.e., $H_{i,*}H_{j,*}^\top = \delta_{i,j}, \forall i, j \in [s]$) with $s \ll d$. For simplicity, we assume that $w - \widehat{w}$ lies in the subspace spanned by the row vectors of H, i.e., $w - \widehat{w} = P_H(w - \widehat{w})$, where $P_H = \sum_{i=1}^s H_{i,*}^\top H_{i,*}$ is a projection matrix. The overall objective function is given by $\min_{w',u}\frac{1}{2}\sum_{i=1}^N\left|x_i^\top w' - y_i\right|^2 + \frac{\alpha}{2}\sum_{i=1}^N\left|x_i^\top w' - x_i^\top\left(\widehat{w} + H^\top \mathrm{sg}(u)\right)\right|^2 + \frac{\beta}{2}\left|y_i - x_i^\top\left(\widehat{w} + H^\top u\right)\right|^2$, where sg denotes the stop gradient. We first examine the convergence of u, whose expression is given as follows

$$\begin{aligned} u &= \left(\sum_{i=1}^N Hx_i x_i^\top H^\top\right)^{-1}\sum_{i=1}^N\left(w + z_i - \widehat{w}\right)^\top x_i Hx_i \\ &= H(w - \widehat{w}) + \underbrace{\left(\sum_{i=1}^N Hx_i x_i^\top H^\top\right)^{-1}\sum_{i=1}^N Hx_i x_i^\top z_i}_{:=v}. \end{aligned} \quad (9)$$

Following the standard concentration inequality, with a probability $1-\delta$, we have $|v| \le O\left(\gamma\sqrt{\frac{s}{N}\log\frac{1}{\delta}}\right)$. Thus $\left|H^\top u - (w - \widehat{w})\right|_2 \le O\left(\gamma\sqrt{\frac{s}{N}\log\frac{1}{\delta}}\right)$. Using the result of u, we obtain the optimal solution as $w_V = w + \frac{\alpha(\widehat{w} + H^\top u - w)}{1+\alpha} + \frac{1}{1+\alpha}\left(\sum_{i=1}^N x_i x_i^\top\right)^{-1}\sum_{i=1}^N x_i x_i^\top z_i$. Following the matrix Bernstein inequality, with an appropriate choice of α, we have, with a probability $1-\delta$, $|w_V - w|_2 \le O\left(\gamma\sqrt{\frac{(ds)^{1/2}}{N}\log\frac{1}{\delta}}\right)$. Compared to the bound for the standard regression model, i.e., $|w_1 - w|_2$, the bound is reduced by the factor $(d/s)^{1/4}$. □