GLAD: A Gating LoRA-Adapter Approach for Efficient Transfer Learning in Recommender Systems

Wentao Hu
University of Science and Technology of China
Hefei, China
huwentao@mail.ustc.edu.cn

Hong Xie*
University of Science and Technology of China
Hefei, China
xiehong2018@foxmail.com

***Abstract*—Leveraging large foundation models for modality-based transferable recommender systems (TransRec)—which learn directly from raw item features like text and images—is a promising paradigm. However, adapting these models via full fine-tuning is resource-intensive and often leads to catastrophic forgetting and overfitting, eroding their valuable pre-trained representations. To achieve parameter-efficient and generalizable TransRec, we propose GLAD (Gating LoRA-Adapter) that adaptively re-weights the influence of Low-Rank Adaptation (LoRA) and Adapter by employing a simple gating module, designed as a learnable controller that generates context-dependent scores from the input representation. This approach transforms the originally static, independent components into a cooperative ensemble, thereby forming an adaptive hybrid strategy, enabling more fine-grained control over the adaptation process. We conduct an extensive evaluation on two textual and one visual recommendation tasks. To assess robustness to data scale, we further evaluate our method on progressively smaller training subsets. While tuning only a small fraction of the parameters compared to full fine-tuning, our method outperforms all baselines on the textual tasks. On the visual task, our approach substantially narrows the performance gap to full fine-tuning, achieving nearly comparable outcomes.**

***Index Terms*—Recommender Systems, Transfer learning, Foundation models**

I. Introduction

The rise of large foundation models [1] is driving significant progress in the development in the field of artificial intelligence. Models such as BERT [2], GPT-3 [3], and Vision Transformers [4] are driving a paradigm shift in Natural Language Processing (NLP) and Computer Vision (CV), leveraging their strong transfer learning performance on diverse benchmark tasks. Inspired by these notable successes, the development of TransRec has emerged as an active research direction [5]. TransRec offers a promising paradigm for enhancing the generalization and adaptability of recommender systems. Its core mechanism involves transferring generalizable knowledge—such as universal user/item representations and their interaction patterns—learned from source domains with abundant data to new or specific recommendation scenarios. This enables superior performance even when the training data in the target context is limited.

Prior to the era of large foundation models, extensive research was dedicated to TransRec, with the majority of approaches being ID-based. Recent efforts such as PeterRec [6], STAR [7] exemplify this category. The modern recommendation models typically represent users and items with unique identities (ID) and map them to learnable embedding vectors. These ID-based recommender models (IDRec) have been established as the widely adopted paradigm in the field of recommender systems for over a decade. However, IDRec suffers from several critical weaknesses that cannot be ignored. Firstly, user and item IDs are generally not shareable across different platforms, which makes it difficult to transfer pre-trained IDRec models to new domains. Secondly, pure IDRec models cannot directly benefit from the technological advancements in other fields, such as NLP and CV, including the powerful foundation models that have emerged in these areas. To enable broader transfer learning— i.e. no longer relying on overlapping and shared-ID information, a prevailing approach is to represent items using their raw modality features (e.g., text or images) and users by their sequence of item interactions, rather than relying on their IDs [8]. By replacing ID embeddings with powerful item modality encoders like BERT and ViT, this generation of TransRec has achieved strong results across numerous downstream tasks. However, the approach of full fine-tuning presents several challenges within the context of TransRec. Primarily, full fine-tuning the entire pre-trained model is computationally intensive. Furthermore, this method may increase the model's risk of overfitting by updating a vast number of parameters, especially when the downstream training data is scarce or exhibits a notable domain shift from the original pre-training task.

To achieve parameter-efficient TransRec, prior work [9] has explored applying standard parameter-efficient transfer learning modules like Adapters. To further explore the effectiveness of this paradigm, we propose GLAD—a method that employs a gating mechanism to dynamically orchestrate LoRA and Adapter, enabling more fine-grained and adaptive

*Corresponding author.

control over the tuning process—and validate its performance and robustness on a more diverse and challenging suite of tasks.

II. Related Work

A. ID-based Recommender Systems (IDRec)

In recommendation scenarios, IDRec has become the predominant methodology. These models capture associative relationships by constructing an interaction matrix from user behaviors, such as clicks and purchases. The evolution of these models has progressed from traditional item-item collaborative filtering [10] and shallow factorization models [11] to complex deep neural networks [12]. They are broadly categorized into two main classes: non-sequential models and sequential recommendation models. Non-sequential models encompass retrieval models like DSSM, as well as Click-Through Rate (CTR) prediction models such as DeepFM [13], Wide & Deep [14]. These models typically process user-item pairs as input to predict a matching score. For interaction data that lacks temporal dependencies, the mainstream approach relies on models based on graph convolutional networks, such as NCL [15]. Additionally, contrastive learning has been incorporated to refine the capture of associations, as demonstrated in models like SimGCL [16].

In contrast, sequential recommendation models leverage sequences of user-item interactions to predict the probability of the next interaction. Representative architectures include the RNN-based GRU4Rec [17], the Transformer-based SASRec [18], and the BERT-based BERT4Rec [19]. Among these, SASRec has been frequently shown to deliver state-of-the-art performance.

B. Modality-based Recommender Systems (MoRec)

Spurred by the notable achievements of foundation models in NLP and CV, research into MoRec has gained significant momentum, aiming to leverage their strong representation learning capabilities [5], [8], [20]. MoRec has emerged as an important research direction to alleviate the classic data sparsity and cold-start problems by leveraging auxiliary side information from items' raw modalities (e.g., texts [21], images [22], videos [23] and text-image multimodal pairs [24]). The central idea is to encode these raw features into semantic representations for recommendation. A notable paradigm shift in MoRec has been the transition towards End-to-End (E2E) training. Many early works adopted a two-stage approach, where features are first extracted using shallow encoders and then fed into a recommendation model. However, these encoders, such as Word2Vec [25], and AlexNet [26], possess limited representation capabilities, failing to capture the deep semantics important for accurate recommendations. In contrast, the recent success of large pre-trained models has enabled the E2E paradigm, where strong encoders like BERT [2] for text and Vision Transformer [4] for images are fine-tuned jointly with the recommendation task. This approach allows the encoders to learn task-specific, high-quality representations. Consequently, numerous recent studies have concentrated on E2E MoRec, with a primary focus on leveraging text modality for news and product recommendation [27].

III. Preliminary

A. Overview of TransRec

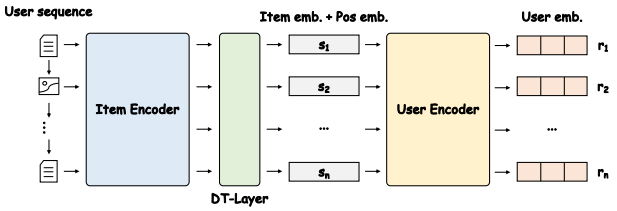

Fig. 1. TransRec framework.

Let $\mathcal{D} = \{\mathcal{U}, \mathcal{V}, \mathcal{I}\}$ denote the recommendation corpus, where $\mathcal{U}$ and $\mathcal{V}$ are the user and item sets, and $\mathcal{I} = \{I_u \mid u \in \mathcal{U}\}$ stores interaction sequences. For any user u, the behavioral history is $I_u = (v_1, \ldots, v_n)$, where n is the length of the interaction sequence.

We aim to predict the next item to be interacted by $u \in \mathcal{U}$ by exploiting users' past behaviors I_u. The TransRec architecture consists of two submodules: the Item Encoder E_{item} and the User Encoder E_{user} (Fig. 1).

1) Item Encoder: For each item $v \in \mathcal{V}$, we generate its vector representation $\mathbf{z}_v$ by encoding its initial raw modality features $\mathbf{m}_v$ using a modality-specific encoder E_{item} (e.g., BERT/RoBERTa for text, ViT/ResNet for images): $\mathbf{z}_v = E_{\text{item}}(\mathbf{m}_v)$.

To align dimensions, a dimension-transformation layer (DTL) is inserted between the two modules, projecting the output of E_{item} into the dimensionality expected by E_{user}: $\mathbf{e}_v = \text{DTL}(\mathbf{z}_v)$.

2) User Encoder: The User Encoder, designed to learn a user's dynamic preferences from their interaction sequence, is typically implemented using a Transformer architecture due to its strong performance on sequential data. However, alternative models such as RNN, GNN, or MLP network can also be employed. Here, We adapt two distinct self-supervised learning frameworks, SASRec and CPC [28], for user representation learning.

The User Encoder generates the representation r_u for user u by processing the interaction sequence I_u. The input representation for each item is the sum of its feature embedding and a corresponding position embedding from $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, and outputs the hidden vectors of corresponding input elements. The specific process is formulated as follows:

$$s_i = \mathbf{p}_i + e_i, \text{for } i = 1, \ldots, n \tag{1}$$

$$(r_1, \ldots, r_n) = E_{\text{user}}((s_1, \ldots, s_n)) \tag{2}$$

where r_i is the representation for user generated from sequence of item interactions up to step i ,e_i are the item vector representations obtained after the items of the user's interaction sequence pass through the Item Encoder, and s_i are then the final input of the User Encoder.

Finally, the resulting user representation vector is used to compute a relevance score, typically via a dot product with the representation of each candidate item. The item with the highest score is then recommended to the user.

B. Parameter-Efficient Transfer Learning (PETL)

Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains. In this way, the dependence on a large number of target domain data can be reduced for constructing target learners. Fine-tuning large pre-trained models is an effective transfer mechanism, however, in the presence of many downstream tasks, full fine-tuning is parameter inefficient. To address this, a parameter-efficient paradigm has emerged, which freezes most of the pre-trained model and only trains a small, task-specific subset of parameters—enabling low-overhead adaptation across dozens or even hundreds of tasks. Among the widely used are Adapter modules, Low-Rank Adaptation (LoRA), Prompt Tuning, and other approaches—each of which injects only a small set of trainable parameters into the frozen backbone.

1) Adapters: Adapters are task-specific neural modules inserted into a pre-trained model which are implemented by embedding compact, auxiliary layers into the structure of Transformer architecture. [29] proposed a parameter-efficient bottleneck module that projects the original features into a lower-dimensional space, applies a non-linear transformation, and then projects them back to the original space. With a residual connection, the module is formulated as:

$$\text{Adapter}(x) = W_{\text{up}}\,\sigma\big(W_{\text{down}}\,x\big) + x. \tag{3}$$

where $W_{down} \in R^{r\times d}$ and $W_{up} \in R^{d\times r}$ represent down-projection and up-projection matrix that project the input dimension up and down, respectively, and $\sigma(\cdot)$ denotes a nonlinear activation function. Pfeiffer Adapter [30] inserts a single bottleneck block per Transformer layer, positioned immediately after the LayerNorm of either the self-attention or feed-forward sublayer, thereby halving the added parameters compared to Houlsby's two-block design. Compacter [31] parameterizes each adapter's projection matrices as the Kronecker product of two shared low-rank bases and employs hypercomplex multiplication layers to dynamically generate the down- and up-projection weights. K-Adapter [32] applies multiple parallel bottleneck adapters within each Transformer layer, each encoding a different type of knowledge, and sums their outputs with the original hidden state via a residual connection before forwarding to subsequent layers of the model.

2) LoRA: LoRA [33] enables parameter-efficient fine-tuning by keeping the pretrained weight $W_0 \in R^{d\times k}$ frozen and injecting only a low-rank update ΔW.This update is factorized into two much smaller matrices $B \in R^{d\times r}$ and $A \in R^{r\times d}$, so that only B and A are optimized during fine-tuning. The module is formulated as:

$$W = W_0 + \Delta W, \quad \Delta W = \frac{\alpha}{r} BA. \tag{4}$$

where r denotes the rank of the low-rank decomposition and α is a scaling factor.

LoRA+ [34] advances the LoRA framework by applying disparate learning rates to adapter matrices ($\eta_B \gg \eta_A$), a technique derived from scaling laws to improve both gradient flow and numerical stability during fine-tuning.

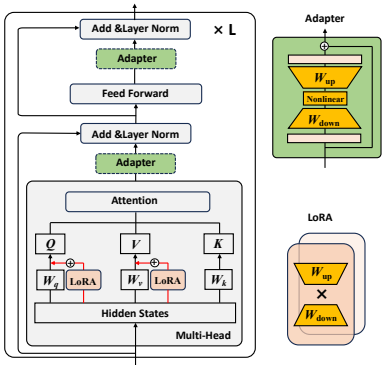

Fig. 2. Transformer architecture with representative state-of-the-art parameter-efficient transfer learning methods.

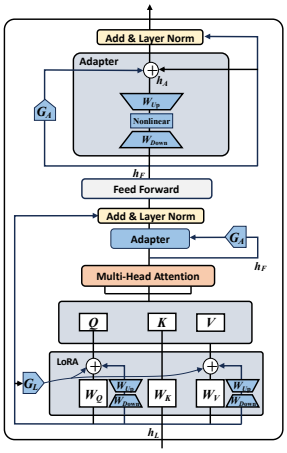

Fig. 3. Overview of the gating mechanism, where blue segments correspond to trainable parameters.

IV. Method

A. Overall

The core principle of PETL is to freeze the vast majority of parameters in a pretrained model, while only training a small subset of newly added or modified parameters. Different PETL methods can be designed to target different functional modules. As shown in Fig. 2, LoRA primarily acts on the weights within the attention layer, directly affecting

the model's sequence relation modeling capability. Adapter, on the other hand, typically functions as a post-processing module, inserted after the multi-head attention layer or feed-forward network to adjust and enrich its output semantic representations. It is the dual difference in operational mechanism and architectural position among different PETL methods that makes the combination of multiple PETL methods possible.

B. Gating Mechanism

To enable fine-grained and adaptive control over the contributions of different submodules, we introduce a trainable gating module for the Adapter and LoRA modules within each Transformer layer, denoted as G_A and G_L (see Fig. 3), respectively. Intuitively, each gating module is designed to dynamically assess the utility of its corresponding submodule conditioned on the input. If a submodule is deemed beneficial for a given instance, its gate will yield a higher value, thereby amplifying the module's influence in the computational graph. Conversely, as the gate value approaches zero, the submodule's contribution is effectively suppressed, tantamount to being bypassed. However, the practical reality is substantially more complex, owing to the intricate interdependencies among these sub-modules and their cumulative effects within deep networks.

Specifically, for the Adapter module, the gating module G_A is implemented as a feed-forward network with a sigmoid activation. It takes the Adapter's input h_F to estimate the module's importance, producing a gating value in the range $(0, 1)$. This value then scales the Adapter's output $h_A = \text{Adapter}(h_F)$:

$$\text{Output}_A = G_A(h_F)h_A + h_F \tag{5}$$

When $G_A \approx 0$, the Adapter submodule is effectively bypassed.

Analogously, we apply a gating mechanism to the LoRA module. Its original design incorporates a static scaling factor, α, to control the contribution of the low-rank path. However, this factor is global and input-agnostic. To introduce adaptive control, we replace this static scalar with a learnable gating module G_L.

Operationally, we set the hyperparameter α to 1, instead allowing the gating module G_L to learn the appropriate scaling. This gate conditions on the input to the LoRA module, h_L, to dynamically modulate the weight of its low-rank path output. Consequently, the forward pass for the entire module is defined as:

$$\text{Output}_L = \left(W_0 + \frac{G_L(h_L)}{r}BA\right)h_L. \tag{6}$$

where $W_0 h_L$ is the original output from the pre-trained weights, and $G_L(h_L)$ is the dynamic gate value computed based on the input h_L.

C. Module Injection Strategy in Encoders: An Asymmetric Approach

A critical design choice in GLAD is not just **what** modules to use, but **where** to inject them. The TransRec framework possesses a natural functional separation between its E_{item} and E_{user}. Recognizing this functional heterogeneity is key to optimizing the transfer learning process. We therefore propose an asymmetric injection strategy, where the two encoders are adapted with different modules.

1) Comprehensive Adaptation for the Item Encoder: The Item Encoder's primary role is to extract detailed semantic representations from raw item content (e.g., text descriptions, images). Its effectiveness relies on the deep, layered feature extraction capabilities of the underlying pre-trained model. When transferring to a new domain, the nature of relevant semantics can shift significantly. For instance, technical jargon in a new domain may require the model to adjust its fundamental attention patterns to focus on different keywords. Simultaneously, how these extracted features are combined into a final item representation also needs to be updated.

To address this, we perform a comprehensive adaptation on the Item Encoder by injecting both Gating LoRA (G-LoRA) and Houlsby-style Gating Adapter (G-Adapter). G-LoRA is injected into the multi-head attention layers to fine-tune the model's focus, allowing it to dynamically attend to new, domain-specific textual or visual cues. G-Adapter is inserted after the feed-forward networks and multi-head attention layers to learn new, high-level feature transformations, re-calibrating the item's semantic meaning for the downstream task. This dual-module approach ensures a holistic adaptation of the item's representation, modifying both low-level attention and high-level processing in a dynamic manner.

2) Preserving Core Sequential Dynamics in the User Encoder: In contrast to the Item Encoder, the User Encoder's primary role is to model the dynamic, sequential patterns of user behavior. Its ability to understand temporal relationships and infer user intent from a sequence of items, learned during pre-training, represents a generalizable form of knowledge that is often directly applicable across different recommendation domains.

Based on this understanding, we posit that altering the core self-attention mechanism (e.g., with LoRA) is unnecessary and risks disrupting these well-learned sequential patterns, potentially leading to negative transfer. Our design principle is therefore to preserve, not reshape, the User Encoder's sequence modeling capabilities.

Consequently, our asymmetric strategy dictates that we only inject G-Adapter into the User Encoder. The adaptation challenge here is not to re-learn sequence modeling, but to teach the model how to interpret the newly adapted item embeddings within a sequence. G-Adapter thus acts as a lightweight, task-specific "interpretation layer," adjusting the model's understanding of user preferences based on the new item representations without corrupting the foundational sequence processing logic.

V. Experiment

A. Experimental Settings

1) Datasets: To evaluate the performance of our proposed method, we conduct experiments on both textual and visual recommendation tasks. For the textual recommendation task,

TABLE I
SUMMARY OF DATASETS.

Dataset	Users	Items	Interactions	Domain
MIND	630,23	79,707	10,928,010	Source
Adressa	82,704	7,659	1,422,408	Target
Eb-NeRD	50,000	7,393	2,074,476	Target
H&M	500,000	86,733	6,500,000	Source
Amazon	21,153	14,348	128,808	Target

we designate the MIND [21] English news recommendation dataset as the source domain. For the target domains, we utilize two distinct non-English news datasets: the Adressa [35] Norwegian news recommendation dataset and the Eb-NeRD [36] Danish news recommendation dataset. A key distinction between these two target datasets lies in their data sparsity and user behavior patterns. The Adressa dataset features an average of 21.88 interactions per user, whereas the Eb-NeRD dataset has a significantly higher average of 246.94 interactions per user. We infer from this statistic that the Eb-NeRD dataset not only possesses greater information density per user but also exhibits higher user heterogeneity, thus posing a more significant challenge for recommendation models. Consequently, to accommodate these differences, we tailor the construction of user interaction sequences: for the Adressa dataset, we use the 20 most recent news articles clicked by each user, while for the more dense Eb-NeRD dataset, we expand the sequence length to the 40 most recent articles. For the visual recommendation task, we employ the H&M personalized fashion recommendation dataset for the source domain and the Amazon Fashion&Shoes dataset [37] as the target domain. Due to GPU memory constraints inherent in processing high-dimensional image features, the user interaction sequence length for this task is set to 10. A summary of the statistics for all preprocessed datasets is provided in Table I.

2) Model Architecture: For item representation, we leverage pre-trained models from the HuggingFace platform as Item Encoder: bert-base-uncased[1] and roberta-base[2] for textual features, and vit-base-patch16-224[3] for visual features. We adapt two distinct self-supervised learning frameworks, SASRec [18] and CPC [28], for user representation learning. Both architectures are optimized using a Binary Cross-Entropy (BCE) loss, tailored to their specific objectives.

In the SASRec framework, the User Encoder is trained on a next-item prediction task. For each user interaction sequence, the model aims to predict the subsequent item based on the representation of the preceding items. This step-wise training objective is formulated as the following loss function:

$$\mathcal{L}_{\text{SASRec}} = -\sum_{u\in U}\sum_{t=1}^{n}\Big[\log\sigma(r_t^u\cdot e_{t+1}^u) + \log(1-\sigma(r_t^u\cdot e_j))\Big]. \quad (7)$$

where r_t^u is the representation for user u generated from sequence of item interactions up to step t, e_{t+1}^u is the embedding of the ground-truth next item (positive sample), and e_j is the embedding of a randomly sampled item from the item corpus (negative sample). The loss is aggregated over all users and all valid positions in their respective sequences.

In contrast, the CPC framework is trained to predict a target item using a context vector that summarizes the entire user interaction history. This encourages model to learn a global representation of user preference. The corresponding BCE loss function is defined as:

$$\mathcal{L}_{\text{CPC}} = -\sum_{u\in U}\left[\log\sigma(r_n^u\cdot e_{n+1}^u) + \log(1-\sigma(r_n^u\cdot e_j))\right]. \quad (8)$$

where r_n^u is the context representation for user u generated from full interaction history. The model is optimized to distinguish the true subsequent item e_{n+1}^u from a negative sample e_j. The key difference from SASRec is that the loss is computed only once per user, based on their complete sequence.

3) Evaluation Metrics: As with the previous works [9], [28], we split the dataset into training, validation, and test sets following the standard leave-one-out strategy . Specifically, for each user, the last interaction is used as the test data, the second-to-last interaction serves as the validation data, and the remaining interactions are used for training. We evaluate all models using two widely-used Top-N ranking metrics: Hit Rate (HR@N) and Normalized Discounted Cumulative Gain (NDCG@N), with N set to 10. We rank the ground-truth target item by comparing it with all the left items in the item pool. Finally, we report results on the testing set.

4) Hyper-parameters: The User Encoder is configured with 2 Transformer layers, 2 attention heads, and a hidden dimension of 64. All models are trained using the Adam optimizer with no weight decay. The learning rate is extensively searched within the range of [5e-6, 4e-3] to find the optimal value for each task. A dropout rate of 0.1 is applied throughout the network to prevent overfitting.

For source domain pre-training, the batch size is set to 64 for textual task and 16 for visual task. During target domain fine-tuning, these are reduced to 16 (text) and 8 (vision). The rank for LoRA modules is fixed at 16, while the hidden dimension for Adapter modules is searched from the set {16, 64, 128, 192, 384}.

B. Performance Comparison

To validate the effectiveness of our proposed GLAD method, we select Houlsby Adapter, Pfeiffer Adapter, K-Adapter, Compactor, LoRA, and LoRA+ as comparative baselines. All methods are evaluated on three datasets. The detailed results are presented in Table II.

[1]https://huggingface.co/google-bert/bert-base-uncased
[2]https://huggingface.co/FacebookAI/roberta-base
[3]https://huggingface.co/google/vit-base-patch16-224

TABLE II

PERFORMANCE COMPARISON ON THE ADRESSA, EB-NERD, AND AMAZON DATASETS. TRAINABLE PARAMETERS FOR EACH MODEL BACKBONE ARE REPORTED WITH THE RESULTS OF THE DATASET WHERE THEY FIRST APPEAR. **BOLD** INDICATES THE BEST RESULT IN EACH ROW.

Architecture	Metrics	Full Fine-Tuning	Houlsby	Pfeiffer	Compactor	K-Adapter	LoRA	LoRA+	GLAD (Ours)
Dataset: Adressa									
SASRec+BERT	HR@10	39.22	38.58	37.30	31.60	35.95	35.76	35.81	**39.43**
	NDCG@10	21.09	21.03	20.01	16.66	19.18	18.65	18.78	**21.50**
CPC+BERT	HR@10	34.89	34.75	34.42	28.43	32.42	32.33	32.51	**35.84**
	NDCG@10	18.44	18.06	18.04	14.99	17.12	17.12	17.21	**19.23**
Trainable Parameters		100%	4.15%	2.12%	0.09%	7.96%	0.56%	0.56%	4.60%
SASRec+RoBERTa	HR@10	38.74	38.03	37.24	32.15	35.76	36.37	36.45	**38.92**
	NDCG@10	20.98	20.67	20.32	16.81	19.11	19.47	19.51	**21.23**
CPC+RoBERTa	HR@10	34.59	34.04	33.66	30.16	31.55	31.50	31.60	**34.72**
	NDCG@10	**18.06**	17.85	17.57	15.20	16.10	16.72	16.79	18.03
Trainable Parameters		100%	3.67%	1.87%	0.08%	7.06%	0.49%	0.49%	4.14%
Dataset: Eb-NeRD									
SASRec+BERT	HR@10	45.08	44.27	43.07	29.39	40.82	40.97	41.08	**45.44**
	NDCG@10	23.72	23.50	22.76	14.23	21.03	21.15	21.25	**24.70**
CPC+BERT	HR@10	37.13	36.25	35.10	25.47	30.35	32.18	32.23	**37.67**
	NDCG@10	18.62	17.93	16.78	12.82	15.38	16.06	16.10	**19.02**
SASRec+RoBERTa	HR@10	45.68	44.71	43.56	31.61	40.75	38.81	38.72	**46.40**
	NDCG@10	24.99	23.64	22.71	15.57	21.19	19.33	19.19	**25.13**
CPC+RoBERTa	HR@10	37.89	37.03	36.71	24.68	34.90	33.85	33.67	**38.25**
	NDCG@10	19.95	19.15	18.92	12.29	18.07	17.66	17.33	**20.03**
Dataset: Amazon									
SASRec+ViT	HR@10	**31.36**	30.11	29.50	23.78	24.08	28.59	28.43	30.62
	NDCG@10	**26.12**	25.08	22.06	15.55	14.25	21.81	21.90	25.88
CPC+ViT	HR@10	**28.71**	27.42	26.55	20.96	19.46	24.12	24.03	28.24
	NDCG@10	22.37	21.47	21.18	12.74	11.34	11.83	11.94	**22.52**
Trainable Parameters		100%	5.24%	2.76%	0.12%	9.93%	0.71%	0.71%	5.91%

The baseline PEFT methods exhibit a clear spectrum of performance-efficiency trade-offs. Houlsby and Pfeiffer deliver decent performance with a relatively small parameter budget, though they do not surpass Full Fine-Tuning. In contrast, K-Adapter, despite utilizing more trainable parameters, shows a drop in performance. While LoRA and LoRA+ achieve modest results, their strength lies in their high parameter efficiency. At the extreme, Compactor, which pursues extreme parameter compression, shows a notable degradation in performance.

GLAD surpasses full fine-tuning in almost all task settings on the two textual recommendation tasks. On the visual recommendation task, although its performance is lower than full fine-tuning, the gap is relatively small, which, as suggested by prior work [9], may be due to the Houlsby Adapter are designed for NLP tasks. Nevertheless, GLAD's results are still the best-performing among all PEFT baselines.

C. Performance under Varying Data Scales

To evaluate the data efficiency and robustness of different methods, we conduct a series of experiments on the Adressa and Eb-NeRD datasets. In these experiments, we vary the dataset size by adjusting the number of users and compare the performance of our GLAD method against full fine-tuning and a high-performing baseline, the Houlsby Adapter. The experimental results are presented in Figure 4.

On the Adressa dataset, we observe that in data-scarce scenarios (e.g., with 10k to 30k users), both GLAD and Houlsby initially outperform the full fine-tuning approach. However,

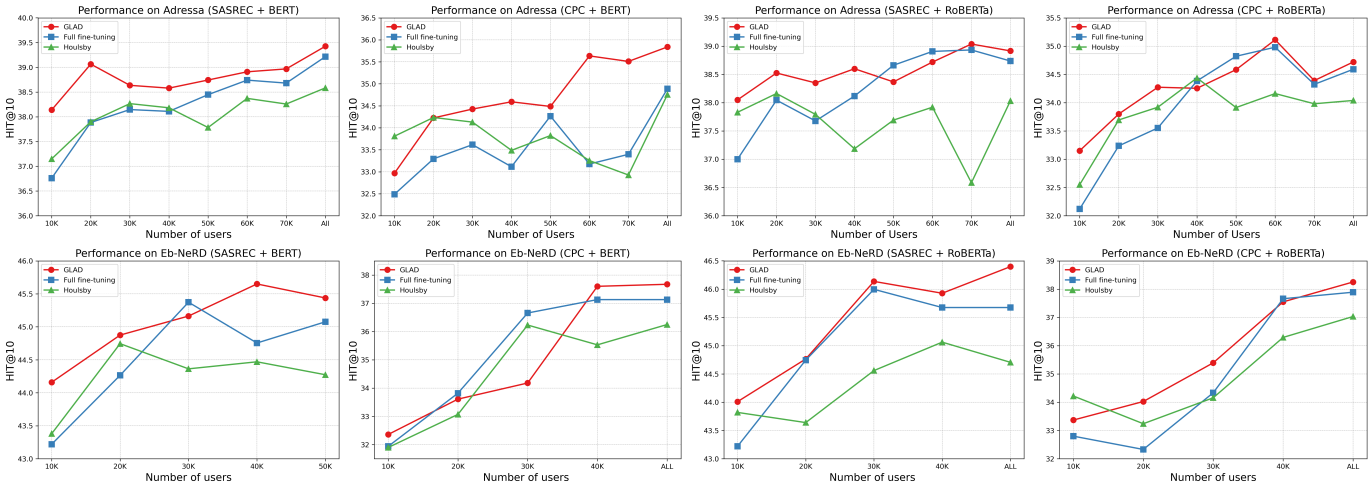

Fig. 4. Performance comparison of different methods under varying downstream data scales.

a clear divergence emerges as the user scale increases. The performance of the Houlsby method is eventually surpassed by full fine-tuning at approximately the 40k user mark. This indicates that full fine-tuning can leverage larger-scale data more effectively than Houlsby, revealing the limitations of the Houlsby method. In contrast, our method maintains its performance advantage under the experimental data scales, outperforming or performing comparably to full fine-tuning. This underscores GLAD's robustness.

This trend is further accentuated on the more challenging Eb-NeRD dataset. The same performance dynamics are observed, but the divergence point appears much earlier. Houlsby's performance declines relative to full fine-tuning after only 20k users, highlighting its limitations in more complex scenarios. Conversely, GLAD once again showcases its consistent superiority, maintaining solid performance regardless of the data scale.

On the more challenging Eb-NeRD dataset, this trend is even more pronounced. The same performance dynamics emerge earlier: after only 20k users, Houlsby's performance is surpassed by full fine-tuning. This highlights its limitations in more complex scenarios. Conversely, our method once again shows consistent and stable performance, maintaining a good showing.

D. Ablation Studies

We conduct a series of ablation studies to deconstruct the GLAD and investigate the efficacy of its two core design principles: the asymmetric injection strategy and the dynamic gating mechanism. The results are presented in Table III.

1) Necessity of the Asymmetric Injection Strategy: The experimental results show that Static-Asymmetric consistently outperforms Static-Symmetric across all tested datasets. This indicates that altering the User Encoder's core attention mechanism interferes with its sequence modeling capabilities and causes negative transfer; thus, preserving its original structure is an effective design choice.

2) Contribution of the Gating Mechanism: Both Static-Asymmetric and GLAD share the same asymmetric injection strategy, differing only in that GLAD incorporates dynamic gating module. The experimental results show that GLAD performs better than Static-Asymmetric. This gain is attributed to the function of the gating mechanism, which enables the model to fine-tune in a more nuanced manner.

3) Module Injection Strategy: On all tested datasets, Houlsby outperforms both Static-Symmetric and Static-Asymmetric, even though the latter two combine different modules and have more trainable parameters. At the same time, our GLAD method outperforms all three of these methods. This shows that a well-designed strategy is more critical than merely increasing parameters, and it further emphasizes that GLAD's performance stems not from a simple stacking of modules, but from the synergy of its asymmetric layout and gating module.

VI. Conclusion

In this study, to address the challenge of efficient transfer learning in recommender systems, we propose GLAD. By leveraging an asymmetric injection strategy and a dynamic gating mechanism, our method enables more fine-

TABLE III
ABLATION STUDIES OF DIFFERENT MODULE INJECTION STRATEGIES. **STATIC-SYMMETRIC** INSERTS LORA AND ADAPTER IN BOTH ENCODERS, WHEREAS **STATIC-ASYMMETRIC** DOES NOT INSERT LORA IN THE USER ENCODER. RESULTS ON ADRESSA AND EB-NERD ARE BASED ON THE SASREC + BERT ARCHITECTURE, WHILE RESULTS ON AMAZON ARE BASED ON THE SASREC + VIT ARCHITECTURE.

Method	Adressa		Eb-NeRD		Amazon	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Houlsby	38.58	21.03	44.27	23.50	30.11	25.08
Static-Symmetric	37.40	20.25	43.63	23.10	28.43	22.09
Static-Asymmetric	38.02	20.61	44.17	23.50	29.63	23.47
GLAD (Ours)	**39.43**	**21.50**	**45.44**	**24.70**	**30.62**	**25.88**

grained control over the transfer process. Extensive experiments demonstrate that this precise control allows our method to strike an important balance between parameter efficiency and performance, ultimately achieving parameter-efficient and generalizable TransRec.

REFERENCES

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

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[5] Y. Cheng, Y. Pan, J. Zhang, Y. Ni, A. Sun, and F. Yuan, "An image dataset for benchmarking recommender systems with raw pixels," in *Proceedings of the 2024 SIAM International Conference on Data Mining (SDM)*. SIAM, 2024, pp. 418–426.

[6] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, "Parameter-efficient transfer from sequential behaviors for user modeling and recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 1469–1478.

[7] X.-R. Sheng, L. Zhao, G. Zhou, X. Ding, B. Dai, Q. Luo, S. Yang, J. Lv, C. Zhang, H. 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*, 2021, pp. 4104–4113.

[8] R. Li, W. Deng, Y. Cheng, Z. Yuan, J. Zhang, and F. Yuan, "Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights," *arXiv preprint arXiv:2305.11700*, 2023.

[9] J. Fu, F. Yuan, Y. Song, Z. Yuan, M. Cheng, S. Cheng, J. Zhang, J. Wang, and Y. Pan, "Exploring adapter-based transfer learning for recommender systems: Empirical studies and practical insights," in *Proceedings of the 17th ACM international conference on web search and data mining*, 2024, pp. 208–217.

[10] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

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

[14] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, 2016, pp. 7–10.

[15] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with neighborhood-enriched contrastive learning," in *Proceedings of the ACM web conference 2022*, 2022, pp. 2320–2329.

[16] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 1294–1303.

[17] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv preprint arXiv:1511.06939*, 2015.

[18] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 197–206.

[19] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441–1450.

[20] J. Wang, Z. Zeng, Y. Wang, Y. Wang, X. Lu, T. Li, J. Yuan, R. Zhang, H.-T. Zheng, and S.-T. Xia, "Missrec: Pre-training and transferring multi-modal interest-aware sequence representation for recommendation," in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 6548–6557.

[21] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu *et al.*, "Mind: A large-scale dataset for news recommendation," in *Proceedings of the 58th annual meeting of the association for computational linguistics*, 2020, pp. 3597–3606.

[22] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[23] Y. Deldjoo, M. Elahi, P. Cremonesi, F. Garzotto, P. Piazzolla, and M. Quadrana, "Content-based video recommendation system based on stylistic visual features," *Journal on Data Semantics*, vol. 5, no. 2, pp. 99–113, 2016.

[24] C. Wu, F. Wu, T. Qi, and Y. Huang, "Mm-rec: multimodal news recommendation," *arXiv preprint arXiv:2104.07407*, 2021.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[27] K. Shin, H. Kwak, K.-M. Kim, M. Kim, Y.-J. Park, J. Jeong, and S. Jung, "One4all user representation for recommender systems in e-commerce," *arXiv preprint arXiv:2106.00573*, 2021.

[28] J. Wang, F. Yuan, M. Cheng, J. M. Jose, C. Yu, B. Kong, Z. Wang, B. Hu, and Z. Li, "Transrec: Learning transferable recommendation from mixture-of-modality feedback," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 2024, pp. 193–208.

[29] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *International conference on machine learning*. PMLR, 2019, pp. 2790–2799.

[30] J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder, "Mad-x: An adapter-based framework for multi-task cross-lingual transfer," *arXiv preprint arXiv:2005.00052*, 2020.

[31] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," *Advances in neural information processing systems*, vol. 34, pp. 1022–1035, 2021.

[32] R. Wang, D. Tang, N. Duan, Z. Wei, X. Huang, G. Cao, D. Jiang, M. Zhou *et al.*, "K-adapter: Infusing knowledge into pre-trained models with adapters," *arXiv preprint arXiv:2002.01808*, 2020.

[33] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[34] S. Hayou, N. Ghosh, and B. Yu, "Lora+: Efficient low rank adaptation of large models," *arXiv preprint arXiv:2402.12354*, 2024.

[35] J. A. Gulla, L. Zhang, P. Liu, Ö. Özgöbek, and X. Su, "The adressa dataset for news recommendation," in *Proceedings of the international conference on web intelligence*, 2017, pp. 1042–1048.

[36] J. Kruse, K. Lindskow, S. Kalloori, M. Polignano, C. Pomo, A. Srivastava, A. Uppal, M. R. Andersen, and J. Frellsen, "Eb-nerd a large-scale dataset for news recommendation," in *Proceedings of the Recommender Systems Challenge 2024*, 2024, pp. 1–11.

[37] R. He and J. McAuley, "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*, 2016, pp. 507–517.