Exploring Adapter-based Transfer Learning for Recommender Systems: Empirical Studies and Practical Insights

Junchen Fu[1], Fajie Yuan[1†], Yu Song[1], Zheng Yuan[1], Mingyue Cheng[2]
Shenghui Cheng[1], Jiaqi Zhang[1], Jie Wang[1*], Yunzhu Pan[1]
[1]Westlake University, [2]University of Science and Technology of China
{fujunchen,yuanfajie}@westlake.edu.cn

ABSTRACT

Adapters, a plug-in neural network module with some tunable parameters, have emerged as a parameter-efficient transfer learning technique for adapting pre-trained models to downstream tasks, especially for natural language processing (NLP) and computer vision (CV) fields. Meanwhile, learning recommender system (RS) models directly from raw item modality features — e.g., texts of NLP and images of CV — can enable effective and transferable recommendations (called TransRec). In view of this, a natural question arises: *can adapter-based learning techniques achieve parameter-efficient TransRec with good performance?*

To this end, we perform empirical studies to address several key sub-questions. First, we ask *whether the adapter-based TransRec performs comparably to TransRec based on standard full-parameter fine-tuning? does it hold for recommendation with different item modalities, e.g., textual RS and visual RS.* If yes, we benchmark these existing adapters, which have been shown to be effective in NLP and CV tasks, in item recommendation tasks. Third, we carefully study several key factors for the adapter-based TransRec in terms of *where and how to insert these adapters?* Finally, we look at the effects of adapter-based TransRec by either scaling up its source training data or scaling down its target training data. Our paper provides key insights and practical guidance on unified & transferable recommendation — a less studied recommendation scenario. We will release all codes & datasets for future research.

KEYWORDS

Recommender System, Parameter-efficient, Transfer Learning, Pre-training & Fine-tuning, Adapter

1 INTRODUCTION

Recently, large foundation models [1] have attracted considerable attention in the entire AI community. BERT [6], GPT-3 [2], ChatGPT [34], CLIP [35] and various Vision Transformers (ViT) [8] have demonstrated impressive transfer learning capabilities on a range of benchmark tasks, and are now reshaping the paradigm of the natural language processing (NLP) and computer vision (CV) communities. Inspired by the enormous success, the research of developing pre-trained & transferable recommendation (TransRec) models is becoming increasingly popular as well [11, 17, 38, 43]. On the other side, the sparsity and insufficient data issues seriously impede the performance of personalized recommender systems (RS). TransRec also provides a natural solution to such problems by performing transfer learning — i.e., transferring knowledge (user/item representations and their matching relations) learned from other large data sources to the current RS tasks with little data.

In fact, research on using TransRec for cross-domain or cold-start recommendation has been ongoing for some time [31, 54]. For example, PeterRec [51, 54], Conure [53], EATNN [3], STAR [37], and NATR [10] applied modern deep neural networks to transfer user- or item-level preference across different recommendation platforms. However, these works are mainly ID-based collaborative filtering (IDRec), which highly relies on overlapped userID or itemID data when transferring knowledge. This strict overlapping assumption hardly holds in practice [26, 55] — e.g., TikTok is unlikely to share their userIDs or itemIDs to YouTube, and vice versa.

To realize more general transfer learning, the common practice [7, 17, 43, 55] is to represent items with their raw modality features (e.g., text or images) and users with the interacted item sequence[1] rather than userIDs and itemIDs. By replacing ID embeddings with powerful item modality encoders (ME), such as BERT and ViT, TransRec have shown state-of-the-art results on many downstream tasks.

Therefore, according to the above works, the modern TransRec framework typically consists of two modules, i.e., a user encoder with one or multiple item ME. TransRec models are often first pre-trained on the large-scale upstream recommendation data and then fine-tuned to adapt a range of downstream recommendation tasks. Here, we argue that the widely adopted full parameter fine-tuning in TransRec has several key issues:

(1) The standard fine-tuning often involves updating the entire pre-trained model. Thereby, in scenarios where RS provides services for multiple vertical channels as described in Figure 1(a), TransRec has to maintain a copy of the fine-tuned model for every channel (as large as the pre-trained model in the main channel). This largely hinders parameter-sharing across domains and brings additional costs in model updates, maintenance, and storage.
(2) The large foundation model could quickly overfit when fully fine-tuning it on a small-scale downstream dataset. Fine-tuning the last (few) layers provide an alternative solution. However, it requires many manual attempts to determine the number of layers to be tuned as it highly depends on the pre-trained model and transfer learning task. This is particularly difficult for TransRec models since they could have multiple item encoders [43].

† Corresponding author. Fajie designed and supervised this research; Junchen performed this research, in charge of key technical parts; Junchen, Fajie, and Yu wrote the manuscript. Yunzhu and Jie collected the Bili and TN dataset, respectively. Other authors assisted in partial experiments.
* The work was done when Jie Wang was a visiting scholar at Westlake University. Her current affiliation is with the University of Glasgow.

[1]From this perspective, the transferable recommendation models in the existing literature are mostly a sequential model.

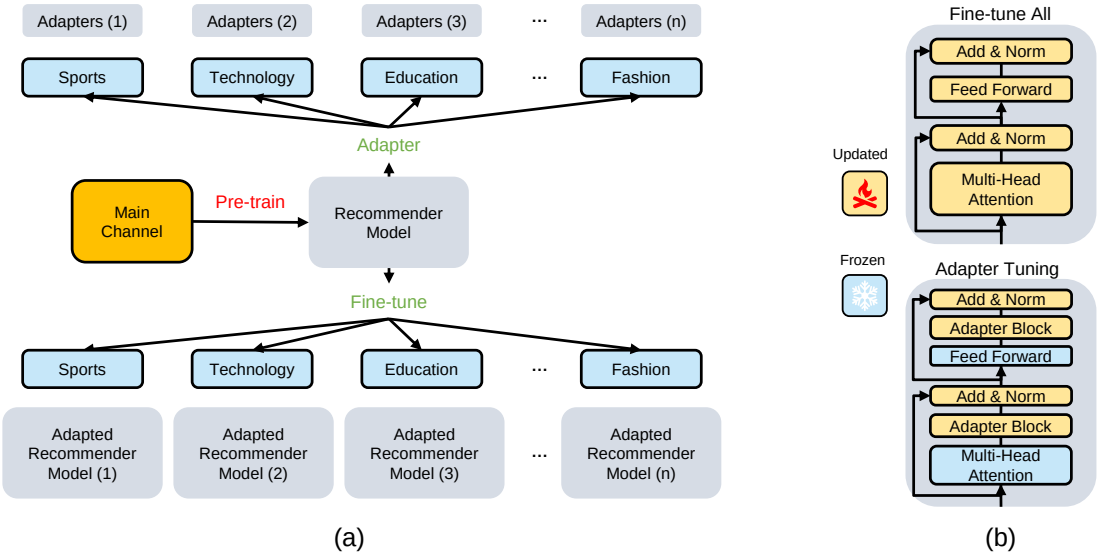


Figure 1: (a) Large industrial RS platforms often have a main recommendation channel and various vertical channels, e.g., sports, education, fashion, etc. One has to maintain the entire model for each channel by a separate model with standard fine-tuning; by contrast, only a small set of parameters need to be maintained by adapter tuning. (b) Fine-tune All (FTA) vs. Adapter Tuning (AdaT). The FTA method updates all parameters in the Transformer [42] block. On the contrary, the AdaT chooses to keep most of the Transformer block frozen and insert new adapter blocks. During training, it only updates these blocks and corresponding layer normalizations.

(3) Fine-tuning the entire ME model could be very expensive and sometimes nearly impractical (e.g., with the billion-level parameter size) in terms of GPU memory. For example, if we use BERT-large (around 1.3G) as the item ME, it typically requires around 10G memory to perform full parameter fine-tuning, e.g., with the item (e.g., news) description sequence length of 50 and batch size of 1, by using only one item ME in TransRec.

These issues of standard fine-tuning motivate us to explore parameter-efficient transfer learning techniques for TransRec. Recent work [19, 41] in NLP and CV suggests that by adding several plug-in networks, i.e., adapter blocks, to the Transformer-style backbones, one can achieve comparable results to full parameter fine-tuning by only optimizing these adapters. Figure 1(b) demonstrates the difference between fine-tuning all parameters (FTA) and adapter tuning (AdaT). Since the number of parameters in these adapters is extremely small compared with the backbone model, they can thereby achieve parameter-efficient transfer learning. For example, by applying the classic Houlsby adapter [19], the number of trainable parameters can be reduced to less than 3% of the entire backbone network. Another advantage of the adapter-based approach is that it enables modular design and easily decouples task-specific parameters from the large backbone network. This mitigates the difficulties of the model maintenance and inconsistent update issues mentioned above. In addition, it can introduce more robustness and achieves improved stability effects during transfer learning as indicated in [13].

Nevertheless, in the recommender system fields, little work has investigated the adapter techniques. A closely related work is PeterRec [51]. However, PeterRec adopts IDRec as the backbone, where the largest amount of parameters is in the ID embedding layer rather than these middle layers in which the adapters are usually inserted. To date, it is still uncertain *whether utilizing adapter-based transfer learning is effective for TransRec models in the process of learning item representations directly from raw modality features.*

To this end, we ask the following sub-questions:

(1) **Q(i) Does the Adapter-based TransRec perform comparably to typical fine-tuning based TransRec? Does this hold for items with different modalities?** To answer it, we conduct a rigorous comparative study for the adapter-based and fine-tuning based TransRec on two item modalities (i.e., texts and images) with two popular recommendation architectures (i.e., SASRec [21] and CPC [43]) and four powerful ME (i.e., BERT, RoBERTa [27], ViT and MAE [14]).
(2) **Q(ii) If Q(i) is true or partially true, what about the performance of these cleverly designed adapters developed in other communities for TransRec problems?** To answer it, we benchmark four adapters widely adopted in the NLP and CV literature. We also add the results of prompt tuning and layer-normalization tuning for a comprehensive comparison.
(3) **Q(iii) Are there any factors that affect the performance of these adapter-based TransRec models?** We report performance comparisons with different strategies regarding how and where to insert the adapters and whether to tune the corresponding normalization layers.

At last, we look at the data scaling effect of TransRec in the source and target domains to examine whether adapter tuning is beneficial when pre-training TransRec with larger datasets.

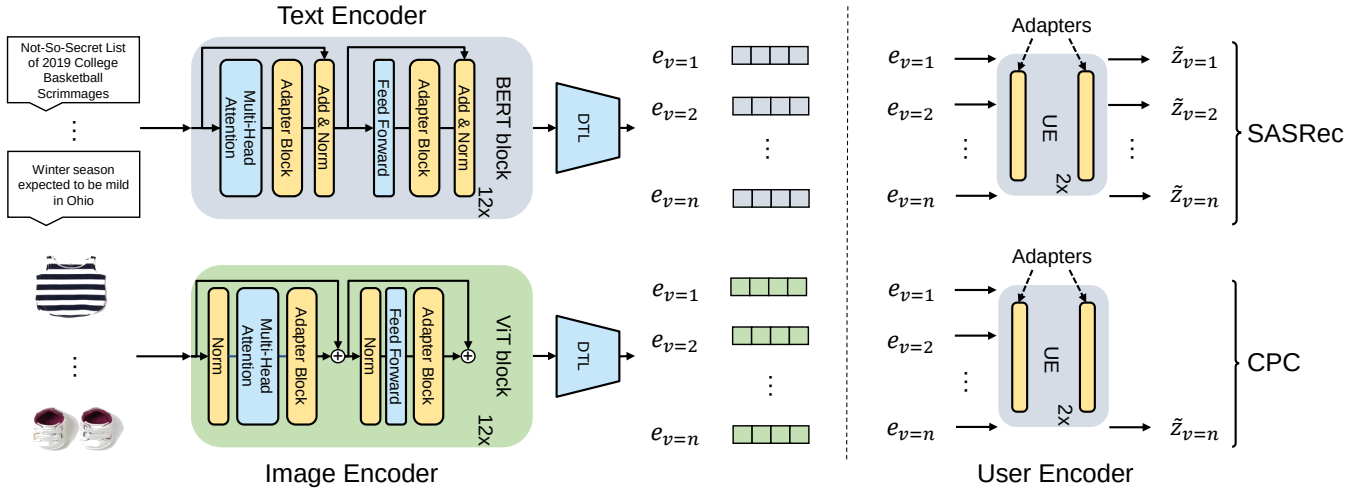


Figure 2: The adapter-based TransRec framework. The TransRec consists of a user encoder (UE) and multiple item encoders divided by the dotted line. BERT and ViT are applied as examples of the text encoder and image encoder respectively. SASRec and CPC (DSSM variant) are used to train UE. $\tilde{Z}_{v=1}$, ..., $\tilde{Z}_{v=n}$ are vector generated by UE, $e_{v=1}$, ..., $e_{v=n}$ are vectors generated by ME. Thereby, the way to inject adapters in UE follows the same way as that of the item encoder.

2 PRELIMINARY

2.1 Overview of TransRec

Given a recommendation dataset $\mathcal{D} = \{\mathcal{U}, \mathcal{V}, \mathcal{I}\}$ where $\mathcal{U}, \mathcal{V}, \mathcal{I}$ denote the set of users, the set of items and the set of interaction sequences, respectively. Like the typical recommendation task, we aim to predict the next item to be interacted by $u \in \mathcal{U}$ by exploiting his/her past behaviors I_u. In TransRec, each item $v \in \mathcal{V}$ is associated with its raw modality features $\boldsymbol{m_v}$. By feeding $\boldsymbol{m_v}$ into an item ME E_{item} (e.g., BERT for text or ViT for images), we obtain the vector representation of item v:

$$\boldsymbol{z_v} = E_{item}(\boldsymbol{m_v}) \tag{1}$$

A basic dimension transformation Layer (DTL) is added to ensure the consistency of the output dimensions of the item ME E_{item} and input dimensions of the user encoder E_{user}:

$$\boldsymbol{e_v} = DTL(\boldsymbol{z_v}) \tag{2}$$

Then, the representation of u can be obtained through the user encoder E_{user} (e.g., a Transformer backbone), which takes the interaction sequence I_u as input:

$$\widetilde{\boldsymbol{z_u}} = E_{user}\left((\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_{|I_u|})\right) \tag{3}$$

Finally, the next item to be interacted by user u can be retrieved from $\mathcal{V}$ by the dot-product similarity between $\widetilde{\boldsymbol{z_u}}$ and $\{\boldsymbol{z_v}\}_{v=1}^{|\mathcal{V}|}$.

As mentioned, we aim to study the transfer learning problem by transferring the knowledge learned from the source domain $\mathcal{S}$ to the target domain $\mathcal{T}$. To be specific, a TransRec model is first trained on the source data $\mathcal{D}_s$ and then adapted to the target domain $\mathcal{T}$, usually by fine-tuning models using target data $\mathcal{D}_t$. Note that $\mathcal{D}_s$ and $\mathcal{D}_t$ do not necessarily contain overlapped items & users. In this paper, we focus on parameter-efficient transfer learning from $\mathcal{S}$ to $\mathcal{T}$ by injecting the task-specific adapters into E_{item} and E_{user}.

2.2 Adapters for TransRec

Adapters overview. Adapters are task-specific neural modules inserted into a pre-trained model. [19] proposed to use a bottleneck network with a few parameters to project the original features to a lower dimension and then project them back after applying a non-linearity. With a residual connection, it can be illustrated as:

$$Adapter(\boldsymbol{y}) = fcUp(RELU(fcDown(\boldsymbol{y}))) + \boldsymbol{y} \tag{4}$$

where $fcUp$ and $fcDown$ represent fully-connected layers that project the input dimensions up and down, respectively. Assuming that the input dimension of the initial model is denoted as d_{model} and the hidden dimension of the adapter is denoted as k, where $k \ll d_{model}$, each transformer block has primary parameters that come from the feedforward network (which contains two fully connected layers with a hidden dimension of d_{ff} that normally equals to $4d_{model}$ [42]) and a self-attention layer that includes the query, key, and value components, as well as one fully connected layer. By implementing two adapters for each transformer block, the total number of trainable parameters can be significantly reduced from $4d_{model}^2 + 2d_{model}d_{ff}$ to $2(2kd_{model})$, regardless of other smaller components such as layer normalization and biases.

Adopting adapters in TransRec. The TransRec architecture contains two sub-modules, namely, the item encoder E_{item} and user encoder E_{user}, both of which are based on the Transformer blocks. The architecture of adapter-based TransRec is illustrated in Figure 2. For E_{item} with textual modality (e.g., BERT), we follow the insertion strategy in [19], where two adapter blocks are inserted into each Transformer block, with one after the multi-head self-attention layer and the other after the feedforward network (FFN) layer. For E_{item} with visual modality (e.g., ViT), the network structure remains the same, except for the position of LayerNorm. The

user encoder E_{user} also uses the same Transformer[2] architecture, the only difference is that E_{user} is unidirectional here. In addition to this, it adopts the same insertion method as the item encoder.

Training objectives. In TransRec, E_{user} takes the interaction sequence of user u (denoted as I_u, with length n) as input, and outputs the hidden vectors of corresponding input elements, i.e.:

$$(\widetilde{\boldsymbol{z_1}}, \widetilde{\boldsymbol{z_2}}, \ldots, \widetilde{\boldsymbol{z_n}}) = E_{user}\Big((\boldsymbol{e_1}, \boldsymbol{e_2}, \cdots, \boldsymbol{e_n})\Big) \tag{5}$$

We use the SASRec [21] and CPC [43] framework to train TransRec. In SASRec, E_{user} is expected to predict the corresponding next item of all elements in I_u, whereas, in the CPC framework, we only aim to predict the (n+1)-th item given the entire sequence. Note SASRec, in general, outperforms CPC in terms of accuracy, but CPC is essentially a more flexible two-tower based DSSM method [43] that is able to incorporate various user and item features. Following [21, 43], we apply the binary cross entropy (BCE) loss for both recommendation frameworks:

$$\begin{cases} -\sum\limits_{u\in U}\sum\limits_{t\in N}\Big[\log\sigma(\widetilde{\boldsymbol{z_t^u}}\cdot\boldsymbol{e_{t+1}^u}) + \log(1-\sigma(\widetilde{\boldsymbol{z_t^u}}\cdot\boldsymbol{e_j}))\Big] & \text{SASRec} \\ -\sum\limits_{u\in U}\Big[\log\sigma(\widetilde{\boldsymbol{z_n^u}}\cdot\boldsymbol{e_{n+1}^u}) + \log(1-\sigma(\widetilde{\boldsymbol{z_n^u}}\cdot\boldsymbol{e_j}))\Big] & \text{CPC} \end{cases} \tag{6}$$

where $\boldsymbol{e_j}$ denotes the embedding of a randomly sampled negative item from $\mathcal{V}$ and $j \notin I_u$. N represents the set of $[1, 2, \cdots, n]$ (see Figure 2).

3 EXPERIMENT SETUP

Datasets. We evaluate adapter-based TransRec with two modalities (text and images). All datasets are divided into two groups: Group 1, with well-known public datasets, is employed for main experiments, whereas due to space limitation, Group 2, with collected private[3] datasets, is only used for a more in-depth examination of adapter tuning for visual recommendation in Section 4. In Group 1, for the textual recommendation, we utilized the MIND [48] English news recommendation dataset as the source domain and the Adressa [12] Norwegian news recommendation dataset as the target domain.[4] As for the visual modality, the H&M[5] personalized fashion recommendation dataset is used as the source domain, and the Amazon dataset for clothing&shoes recommendation [29] is used as the target domain. Group 2 includes a collection of user behavior datasets focused on image-based recommendation from several popular video RS platforms. This collection includes a large-scale source domain dataset from Bili's main channel (Bili_MC) and six downstream datasets. Three of these are gathered from Bili's vertical channels (Bili_F, Bili_C, and Bili_D),[6] while the remaining three are collected from separate platforms (TN, DY, and KS).[7] Note that there are no overlapped items or user-item interactions between Bili_MC and Bili_*. A few users may visit both the main and vertical channels, but overlapping users are not our focus. We select the latest 20 clicked items to construct interaction sequences for text recommendation tasks. Due to the constraint of GPU memory, the sequence length for image recommendation is limited to 10. After the pre-processing, the details of the datasets are shown in Table 1.

Evaluations. Following previous works [16], we adopt the leave-one-out strategy to split the datasets: the last item in the interaction sequence is used for evaluation, and the item before the last is used as validation while the rest are for training. The HR@10 (hit ratio) [51] and NDCG@10 (Normalized Discounted Cumulative Gain) [43] are used as evaluation metrics. Without special mention, all results are for the testing set. Note that we rank the predicted item with all items in the item set [23].

Implementation Details. The "bert-base-uncased", "roberta-base", "vit-base-patch16-224", and "vit-mae-base" from the Huggingface platform[8] are used as the text and image encoders, respectively. The dimension of hidden representations of the user encoder is searched in {32,64,128} and set to 64. and the number of Transformer blocks and attention heads is fixed to 2. We apply Adam as the optimizer without weight decay throughout the experiments and extensively search the learning rate from 1e-6 to 1e-2 while keeping the dropout probability to 0.1. We set the batch size to 64 for textual datasets and 32 for visual datasets due to the GPU memory limits. When adapting to the target domain, we set the batch size to 32 for both modalities. The hidden dimensions of the adapter networks are carefully searched in {8, 16, 32, 48, 64, 96, 128, 192, 384, 768}, and the number of tokens of prompt tuning in {5, 10, 20, 30, 40, 50}. Note that the hyper-parameters of parameter-efficient modules are only searched in the SASRec-based architectures and directly transferred to the CPC-based methods. All hyper-parameters are determined according to the performance in the validation data. All results are reported on the testing set.

4 EFFECTIVENESS OF ADAPTERS IN TRANSREC (Q(I))

In this section, we evaluate the effectiveness of adapter tuning (AdaT) in TransRec since it is unknown whether AdaT works or not for recommendation models. Specifically, we run experiments on eight combinations: {SASRec+BERT, CPC+BERT, SASRec+RoBERTa, CPC+RoBERTa, SASRec+ViT, CPC+ViT, SASRec+MAE, CPC+MAE}, where BERT, RoBERTa, ViT, and MAE are the most

[2]One might wonder whether other networks can be used as TransRec backbone. In fact, TransRec that learns recommendation models directly from item *raw* modality features (vs. ID features, vs. pre-extracted fixed features [18] from ME) is still at a very early stage. Several existing literature [18, 36, 39, 43] is all based on the Transformer-style backbone, the most well-known SOTA sequential encoder. In practice, the Transformer backbone can be easily replaced with other sequential networks. Second, can the CTR (click-through rate prediction) models be used as TransRec backbones? Unfortunately, the classical one-tower CTR models (e.g., DeepFM [28] & MMOE [28]) cannot be directly used as a pre-training backbone for TransRec since some domain-specific features are not transferable or easily decoupled when adapting to other datasets. Instead, the two-tower DSSM model [46] can often be used to pre-train TransRec, as shown below.

[3]All datasets used in this paper will be released after acceptance.

[4]We translate the Adressa dataset from Norwegian to English using Google Translate and randomly sample 20,000 users to construct the dataset.

[5]https://www.kaggle.com/competitions/h-and-m-personalized-fashionrecommendations/overview

[6]The Food, Cartoon, and Dance vertical channel dataset are denoted by Bili_F, Bili_C, and Bili_D, respectively

[7]Bili: https://www.bilibili.com/; TN: https://news.qq.com/; KS: https://www.kuaishou.com/new-reco; DY: https://www.douyin.com/. Group 2 datasets were collected based on public comments. To collect these datasets, we conducted a random crawl of short video (lasting less than 10 minutes) URLs across corresponding video channels. Subsequently, we retrieved the public comments on these videos as interactions. Ultimately, we combined all user interactions in chronological order and removed instances of repeated interactions.

[8]https://huggingface.co/

Table 1: Dataset Description

Dataset	Users	Items	Interaction	Content	Domain	Group
MIND	630,235	79,707	10,928,010	Text	Source	Group 1
Adressa	20,000	3,149	280,656	Text	Target	Group 1
H&M	500,000	86,733	6,500,000	Image	Source	Group 1
Amazon	21,153	14,348	128,808	Image	Target	Group 1
Bili_MC	500,000	137,493	5,976,285	Image	Source	Group 2
Bili_F	6,549	1,576	39,285	Image	Target	Group 2
Bili_C	30,300	4,722	206,165	Image	Target	Group 2
Bili_D	10,715	2,302	74,885	Image	Target	Group 2
TN	20,293	3,765	132,914	Image	Target	Group 2
DY	20,398	8,293	136,321	Image	Target	Group 2
KS	2,034	4,836	14,264	Image	Target	Group 2

popular and widely accepted state-of-the-art (SOTA) ME in NLP and CV fields.

The most prevalent AdaT — i.e., Houlsby [19] Adapter — results are present in Table 2. Note that other adapter results are reported in the next benchmark section. As can be seen, TransRec, with the SASRec sequence-to-sequence (seq2seq) training approach, consistently outperforms its CPC version with sequence-to-one (seq2one) approach. This is perhaps because the seq2seq training approach is more powerful at modeling the item transition pattern in the user sequence and can correspondingly alleviate the insufficient training data issue. For the text recommendation task, AdaT yields comparable results to fine-tuning all parameters (FTA) across evaluated frameworks (SASRec/CPC+BERT/RoBERTa) with a parameter reduction rate of over 97%. However, for image recommendation, the performance gap between FTA and AdaT is relatively large, regardless of the training strategies used (SASRec/CPC+ViT/MAE). This result is somewhat justified, as the Houlsby adapter is primarily designed for NLP domain data and scenarios, which may make it suboptimal for visual tasks.

To gain a deeper understanding of AdaT for image recommendation, we carried out additional experiments on six downstream datasets from Group 2, encompassing three cross-domain and three cross-platform datasets,[9] where the latter is generally a more challenging transfer learning task. As illustrated in Figure 3a, AdaT performs consistently well in cross-domain transfer learning but not as well as FTA in cross-platform settings. This aligns with our previous findings for visual recommendation in Group 1 datasets. These findings demonstrate AdaT's effectiveness in image recommendation for cross-domain scenarios but also indicate the ongoing challenges in cross-platform scenarios.

To understand the impact of trainable parameters on domain adaptation, we simply test AdaT with different adapter sizes and compare its performance with top-n layer fine-tuning (FTN) for text recommendation. Since most of the trainable parameters come from item ME, we focus on the adapters in item ME and keep the UE with the original settings. The results are shown in Figure 3c, where the x-axis denotes the number of trainable parameters that are changed by gradually increasing/decreasing the hidden dimension of the adapter module (for AdaT) or by tuning more/fewer top Transformer layers (for FTN). Clearly, both FTN and AdaT can improve performance with more trainable parameters. Furthermore, AdaT achieves competitive results with nearly two orders of magnitude fewer parameters than FTN.

Another key advantage of AdaT is its robustness to the learning rate during training. In Figure 3b, we demonstrate the HR@10 accuracy of text recommendation task with learning rates across two orders of magnitude, i.e., [1e-5, 1e-4, 1e-3]. AdaT achieves decent recommendation accuracy at all three learning rates, whereas FTA fails when the learning rate is too large.

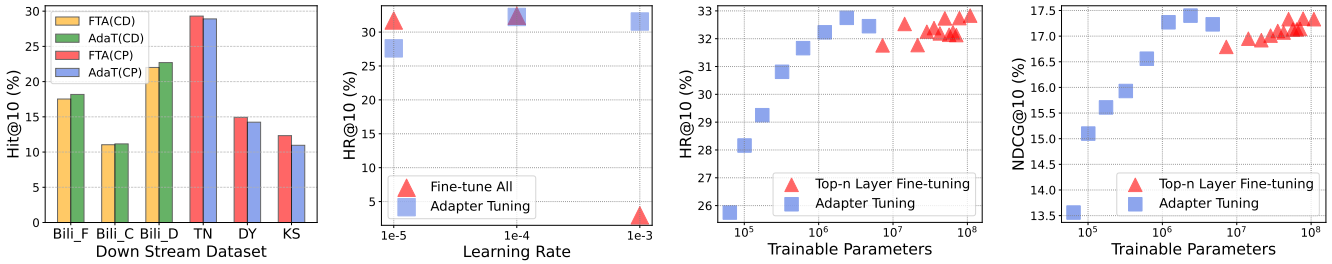
(Answer for Q(i)) Overall, when learning items with textual content, TransRec with the SOTA AdaT achieves the parameter-efficient transfer from the source to the target domain, yielding comparable performance to FTA. However, AdaT improves parameter efficiency in the cross-platform recommendation setting at the cost of some accuracy drop for visual recommendations but is still a viable option for cross-domain scenarios, as depicted in Figure 1(a).

5 BENCHMARKING PARAMETER-EFFICIENT METHODS (Q(II))

In this section, we go one step further and benchmark four popular adapters in NLP and CV literature for applications in recommender systems. To be specific, we choose the Houlsby adapter [19], the K-Adapter [44], the Pfeiffer Adapter [32], and Compacter [22] for evaluation. For a comprehensive comparison, we also include the results using prompt tuning [20] and LayerNorm tuning [4]. We report the results in Table 3.

The general structures of Houlsby, Pfeiffer, Compacter, and K-Adapter are illustrated in Figure 4. The Pfeiffer architecture only inserts one adapter block for each Transformer block, saving about half of the parameters of the Houlsby. We adopt the implementation of the Pfeiffer adapter in [22]. The Compacter is constructed upon low-rank optimization and parameterized hypercomplex multiplication layers. The K-Adapter adds the adapters of Transformer structures to the backbone model in parallel. We insert two adapters for the item and user encoder respectively after structure searching. We also evaluate a popular prompt tuning [24] technique as the baseline method, where the newly inserted token embeddings are

[9] Cross-domain recommendation means items are from the same platform but across different channels.



(a) More scenarios for visual RS (b) Learning Rate Adaptation (c) Adapter tuning vs. top layer tuning.

Figure 3: (a) Cross-Domain and Cross-Platform Transfer Learning in Visual RS. The presented results are on SASRec+ViT. CD and CP denote Cross-Domain and Cross-Platform, respectively. (b) *Robustness* with different learning rates. The three learning rates, 1e-5, 1e-4, and 1e-3, are adopted for both FTA and AdaT. (c) Top-n layer fine-tuning optimizes the top n layers ($n = 1, 2, \cdots, 12$). Adapter tuning with an adapter size of 2^n ($n = 0, \cdots, 7$).

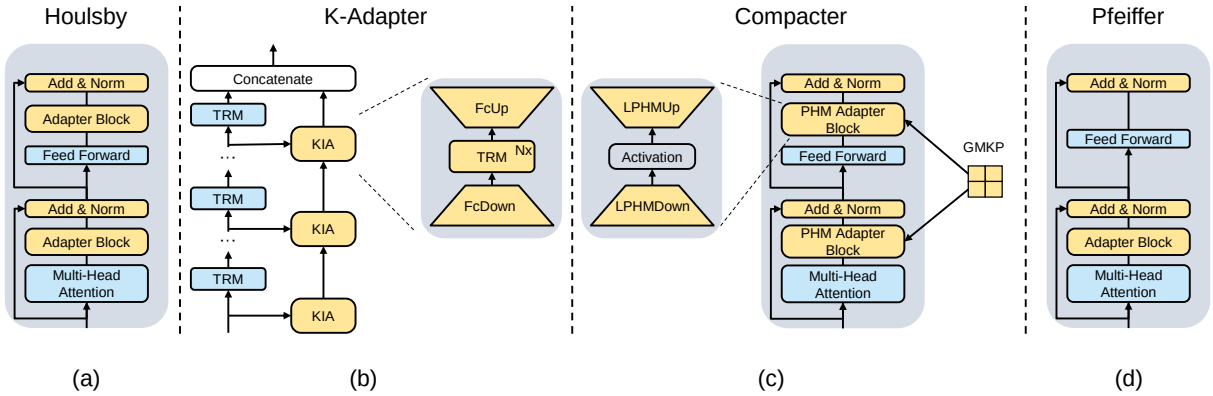
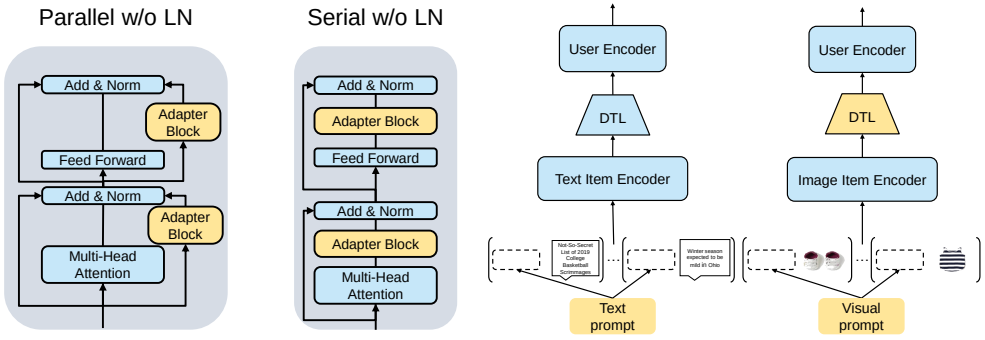


Figure 4: Structure of different adapters implemented in this paper. (a) The Houlsby adapter adopts a bottleneck network to project the intermediate outputs up and down, as described in Section 2. (b) The K-Adapter replaces the activation function in the bottleneck network with N=2 Transformer layers (i.e., KIA in the figure). (c) In Compacter, the FC layers are replaced by a low-rank parameterized hypercomplex multiplication layer (LPHM). GMKP stands for Global Multiplier of the Kronecker Product, which is shared among all Transformer blocks in the backbone model. (d) The Pfeiffer architecture differs from the Houlsby version in that it only adds one adapter block to the Transformer layer.



(a) Adapter Tuning (b) Prompt Tuning

Figure 5: (a) Different insertion methods for the Houlsby Adapter. We present the parallel and serial structure without LayerNorm tuning. (b) Prompt tuning for Text- and Image-based scenarios. In our experiments, we tune the DTL in the visual recommendation and freeze it in the textual scenario.

Table 2: Fine-tuning and adapter tuning comparison. FTA and AdaT represent "Fine-tune All" and "Adapter Tuning" respectively. All results of HR@10 and NDCG@10 in this table are denoted in the percentage (%). Text and Image represent the textual and visual recommendation. The arrow "->" symbolizes the "transfer".

Datasets	Architecture	Metrics	FTA	AdaT	Difference
MIND->Adressa	SASRec+BERT (**Text**)	HR@10	**32.83**	32.52	-0.94%
		NDCG@10	17.33	**17.44**	+0.63%
	CPC+BERT (**Text**)	HR@10	29.56	**30.07**	+1.69%
		NDCG@10	15.81	**16.12**	+1.92%
		Trainable Parameters	100%	2.23%	-97.77%
	SASRec+RoBERTa (**Text**)	HR@10	32.02	**33.14**	+3.38%
		NDCG@10	16.95	**17.54**	+3.36%
	CPC+RoBERTa (**Text**)	HR@10	29.90	**30.64**	+2.42%
		NDCG@10	15.86	**16.20**	+2.10%
		Trainable Parameters	100%	1.95%	-98.05%
H&M->Amazon	SASRec+ViT (**Image**)	HR@10	**29.00**	27.66	-4.59%
		NDCG@10	**25.61**	24.36	-4.88%
	CPC+ViT (**Image**)	HR@10	**26.56**	25.29	-4.78%
		NDCG@10	**22.09**	21.49	-2.72%
		Trainable Parameters	100%	2.82%	-97.18%
	SASRec+MAE (**Image**)	HR@10	**28.10**	25.67	-8.61%
		NDCG@10	**22.92**	21.99	-4.05%
	CPC+MAE (**Image**)	HR@10	**27.50**	25.18	-8.44%
		NDCG@10	**23.51**	21.83	-7.14%
		Trainable Parameters	100%	2.82%	-97.18%

Table 3: Benchmark popular parameter-efficient tuning techniques. Houlsby, K-Adapter, Pfeiffer, and Compacter adapters, along with LayerNorm (LN) and prompt tuning, are presented. The best approach to each architecture is marked in bold in this section. The "Architecture" is the combination of the user and item encoder. All results of HR@10 and NDCG@10 in this table are denoted in the percentage (%). We represent the trainable parameters of each method by the percentage to the full fine-tuning. Due to space limitations, we omit the results of RoBERTa and MAE as ME, which are consistent with BERT & ViT.

Architecture	Metrics	Houlsby	Pfeiffer	Compacter	K-Adapter	Prompt tuning	LN-tuning
SASRec+BERT	HR@10	**32.52**	31.49	29.56	31.30	20.42	15.85
	NDCG@10	**17.44**	16.62	15.32	16.71	11.02	7.89
CPC+BERT	HR@10	**30.07**	29.32	26.40	28.49	20.31	15.81
	NDCG@10	**16.12**	15.37	13.73	15.01	10.42	8.65
	Trainable Parameters	2.23%	1.13%	0.09%	8.69%	0.03%	0.04%
SASRec+ViT	HR@10	**27.66**	27.34	21.77	22.17	22.10	19.89
	NDCG@10	**24.36**	21.45	14.81	13.65	15.72	13.49
CPC+ViT	HR@10	**25.29**	24.63	18.91	18.05	19.43	18.79
	NDCG@10	**21.49**	21.37	11.86	11.36	13.99	13.00
	Trainable Parameters	2.82%	1.43%	0.12%	11.03%	0.07%	0.05%

added to the word embedding layer in the BERT model, as illustrated in Figure 5b. For visual recommendation, VPT [20] is used as the prompt tuning method, which adds the new token patch in the positional patch embedding for the ViT model. VPT needs to update the task-specific head compared to prompt tuning for text. We update the *DTL* module as the task-specific head following the original setup (see Figure 5b).

The Houlsby adapter, among all methods, yields the best results under all settings with less than 3% of trainable parameters of full fine-tuning. Following Houlsby, Pfeiffer achieves close performance with only around half of the parameters. This is because their adapter architectures are similar. The key difference is Pfeiffer removes adapters after FFN (see (a&d) in Figure 4). The reason why Pfeiffer performs relatively worse will be further discussed in Section 6. The conclusion is that the position of adapter blocks does affect the overall performance.

Compacter, with a special focus on parameter compression with low-rank factorization methodology, exhibits a significant decrease in recommendation accuracy, especially in image-based tasks. This is most likely due to the extremely low capacity of trainable modules in Compacter. Figure 3c also shows that reducing parameters by a large amount can lead to very bad results. Therefore, TP (trainable parameters) matters in a certain range in the recommendation task.

Table 4: Comparison of full adapter-based TransRec and only adding adapters to the item or user encoder. Adapter$_{E_i}$ and Adapter$_{E_u}$ denote only adding adapters to the item and user encoder respectively. TP stands for trainable parameters.

Architecture	Metrics	Adapter	Adapter$_{E_i}$	Adapter$_{E_u}$
SASRec+Bert	HR@10	**32.75**	32.45	3.17
	NDCG@10	17.40	**17.50**	1.58
CPC+Bert	HR@10	**30.07**	29.56	17.49
	NDCG@10	**16.12**	16.02	9.34
SASRec+ViT	HR@10	**27.89**	15.79	5.78
	NDCG@10	**24.67**	10.51	1.75
CPC+ViT	HR@10	**25.48**	23.37	9.57
	NDCG@10	**21.73**	19.32	6.57
	TP	100%	99.64%	0.36%

Table 5: Adapter position impact inside a Transformer block. We present the HR@10 for text and image recommendation with SASRec+BERT and SASRec+ViT architectures. Adapter$_{FFN}$ and Adapter$_{MHA}$ represent inserting the adapter block after FFN and MHA respectively. And Adapter$_{MHA}$++ and Adapter$_{FFN}$++ stand for the same architectures as the previous two but with 2x the parameters.

Method	Text	Image	TP
Adapter$_{MHA+FFN}$ (Houlsby)	**32.52**	**27.66**	2,426,816
Adapter$_{MHA}$ (Pfeiffer)	31.49	27.34	1,232,928
Adapter$_{FFN}$	31.72	27.01	1,232,928
Adapter$_{MHA}$++	31.71	27.49	2,417,472
Adapter$_{FFN}$++	31.58	27.05	2,417,472

One exception here is the K-Adapter, which adopts a Transformer layer within the adapter module and requires much more parameters to train than its counterparts. Surprisingly, the performance drops severely. We conjecture that the Transformer architecture within the K-Adapter is not suitable for domain adaptation since it is originally designed for knowledge injection rather than parameter-efficient purposes, see Figure 4b. K-Adapter does not inject its information into the pre-trained model. Instead, it only receives knowledge from pre-trained models. The information flow direction makes its working mechanism very different.

The last two columns in Table 3 show the results for prompt tuning and LayerNorm tuning. Prompt tuning offers a flexible way to utilize a big pre-trained model in various downstream tasks, mainly in the NLP domain. However, it fails to give competitive results as the adapters in TransRec. LayerNorm tuning, i.e., only updating the LayerNorm parameters during adaptation, also suffers from severe performance degradation. These results again potentially imply that TP are important for recommendation, although many NLP tasks can be performed well even with much fewer TPs.

(Answer for Q(ii)) Overall, the Housby adapter obtains the best results in TransRec under all experimental settings, while the Pfeiffer adapter achieves slightly worse performance with half the number of parameters. In the domain of NLP, Compacter yields significantly better performance than the popular Houlsby and Pfeiffer adapters, even with an extremely small amount of trainable parameters [22]. However, it fails to achieve decent results in the modality-based recommendation task. The LayerNorm tuning routinely performs worse than the full fine-tuning and adapter techniques in both CV and NLP with an accuracy drop of about 10% to 20% [15, 19]; however, it is only half as accurate as the best Houlsby method in the recommendation scenarios. **One key finding is that the adapter's trainable parameter size, insertion positions, and information flow directions are all key factors for the recommendation task.**

6 ANALYSIS OF MORE FACTORS (Q(III))

Since existing adapters are mainly derived from the NLP literature, a natural challenge is to effectively apply them in the recommendation scenario. Specifically, we ask: *where and how to insert the adapters for TransRec?* Regarding the question of *where*, we aim to check whether the two modules in TransRec, E_{item} and E_{user}, are equally important for domain transfer, as this is specific for the recommendation task. Regarding the question of *how*, we evaluate two insertion strategies (serial vs. parallel) of the adapter networks and explore the effect of LayerNorm in the recommendation task.

We first evaluate the effect of adapters inserted into different modules in TransRec. There are three ways to implement adapters: placing them into both user and item encoders ($AdaT_{all}$), only into the item encoder ($AdaT_{item}$), and into the user encoder ($AdaT_{user}$) (all other parameters are fixed). From Table 4, first, we can clearly see that $AdaT_{item}$ outperforms $AdaT_{user}$ by a large margin in all experimental settings. This indicates that the item encoder plays a more important role in the recommendation task and requires more re-adaptation on the new datasets. Second, $AdaT_{item}$ achieves comparable results as $AdaT_{all}$ in textual RS, suggesting that the knowledge stored in E_{user} can be largely re-used with the adapted E_{item}. However, there is still a significant gap between $AdaT_{item}$ and $AdaT_{all}$ for the visual task, indicating that the parameter adaptation of E_{user} is also important. Besides, This again shows that the image-based visual recommendation is a more difficult task than the text recommendation.

From Section 5, we know that the Pfeiffer adapter shows a performance gap from Houlsby. The only difference is that Houlsby inserts adapter blocks after both FFN and MHA, whereas Pfeiffer only inserts after MHA. We further test the setting of this adapter to verify the impact of the position of insertion. The results are in Table 5. Thereby, accuracy drops may come from two reasons: 1) no tunable adapter after FFN; 2) fewer TP because of removing one adapter. To verify 1), we changed the position of the adapter (i.e., inserting after FFN), which yielded the same results. To verify 2), we double the number of TP, which still performs similarly, indicating adapters should be inserted for both FFN and MHA for RS.

We then compare two adaption insertion strategies: serial and parallel, as shown in Figure 5a. The parallel approach is also adopted in [19, 57]. In Table 6, it can be seen that the two insertion methods, in general, perform very similarly. The other observation is that whether tuning the LayerNorm layer or not has almost no obvious influence on the recommendation accuracy. This is very different from other fields [19, 22] where they strongly suggest optimizing

Table 6: Performance of the adapter insertion methods. We present the performance of adapter insertion methods for text and image recommendations. The methods include both serial and parallel insertion, with and without tuning the LayerNorm. 'w/' and 'w/o' denote with and without LayerNorm updating. Subscripts $_B$ and $_V$ represent BERT and ViT, respectively.

Method	LayerNorm	SASRec$_B$	CPC$_B$	SASRec$_V$	CPC$_V$
Serial	- w/ LN	32.52	30.07	27.66	25.30
	- w/o LN	**32.75**	29.62	**27.89**	**25.48**
Parallel	- w/ LN	32.70	**30.28**	26.63	24.92
	- w/o LN	31.88	29.66	27.39	24.92

both the adapter and LayerNorm layers for obtaining the optimal results. Thereby, for practical recommendation tasks, we only need to save the adapter modules, which is more efficient and convenient.

(Answer for Q(iii)) We draw some conclusions here: (1) TransRec should place adapters for both user and item encoders for obtaining the optimal results, in particular for visual RS whose performance drop is very significant if the parameters of either the item encoder or user encoder are completely fixed; (2) the insertion position on the Transformer layers is also important, both FFN and MHA require a separate adapter module; (3) other factors such as insertion way (serial or parallel) and LayerNorm optimization do not matter a lot for the recommendation task, although they are often considered for NLP and CV tasks; (4) again, the number of trainable parameters is always a key factor for the accuracy of TransRec, certainly within a certain range, as described in the previous section.

7 SCALING EFFECTS

To better understand the role of training data during pre-training and downstream adaption, we conduct experiments by scaling data in both the source domain ($\mathcal{D}_s$) and the target domain ($\mathcal{D}_t$), and present the results in Figure 6. According to the performance curves, we make the following observations:

(1) Despite some exceptions, the HR@10 shows a clear trend of improvement for FTA and AdaT on the two modalities as the size of upstream pre-training dataset in $\mathcal{D}_s$ increases. This observation has important implications: **for industrial recommender systems, one can expect greater accuracy gains with a larger pre-trained source domain dataset.**

(2) IDRec is often a strong baseline for the recommendation task when there is sufficient training data. Here we compare AdaT to IDRec by scaling the target domain dataset. Clearly, as the size of the downstream datasets (Adressa and Amazon) in $\mathcal{D}_t$ gets larger, the relative improvement of TransRec (FTA and AdaT) over IDRec becomes smaller. In other words, **TransRec (with AdaT) is a preferable option when the target domain has very limited data, such as in a newly established system.**

(3) AdaT shows poor results under the NoPT setting, where only the item ME is pre-trained (on some NLP and CV data, e.g., ImageNet), and the user encoder is randomly initialized. This explains that the lightweight adapter network indeed (or can only) does some parameter adaption work. It should fail or perform worse when the parameters (in the user encoder) are randomly initialized.

(4) There are some other observations consistent with the previous description. For example, AdaT achieves comparable results to FTA for the text-based recommendation, while it lags behind FTA for image-based recommendation in the cross-platform scenario regardless of the size of the source and target datasets. We omit repeated descriptions here.

8 RELATED WORK

Parameter-efficient transfer learning (PETL). Researchers have been working on PETL for years to alleviate the gigantic amount of trainable parameters in large-scale pre-trained models. The principal way is to introduce adapter tuning techniques [33]. In NLP, the first adapter was proposed in [19] where authors uncovered that only training the newly inserted adapter blocks without any modification of the pre-trained parameters could achieve competitive results to full parameter fine-tuning. Pfeiffer et al. [32] proposed the AdapterHub framework to facilitate, simplify, and speed up transfer learning across a variety of languages and tasks. [44] proposed to inject multiple kinds of knowledge into large pre-trained models by K-Adapter. He et al. [15] explored the PETL techniques in ViT-based computer vision tasks. The recent preprint paper, ViT-Adapter [4] achieved state-of-the-art performance on many dense prediction tasks of CV.

Prompt [25, 45, 56] is another popular PETL paradigm. It shows that only optimizing the embeddings of a few prompt tokens exhibits similar performance as the full model fine-tuning. Recently, P5 [11] presented a "pretrain, personalized prompt & predict paradigm" that can learn multiple recommendation-related tasks together by formulating them as prompt-based natural language tasks. M6-Rec [5] showed that prompt tuning outperformed fine-tuning with negligible 1% task-specific parameters. However, in this paper, with the two popular TransRec architectures, we found that the standard prompt tuning is still unsatisfactory compared to adapter- or fine- tuning. [49] and [50] utilized prompt tuning to study the selective fairness and cold-start recommendation, but are ID-based methods different from our modality setting.

Modality-based TransRec. Inspired by the success of foundation models in NLP and CV fields, the modality-based/only recommendation (MoRec) has attracted rising attention recently [26, 43, 55]. Typically, they use the foundation model, such as BERT, RoBERTa, and GPT [2] as the text encoder or ViT, and ResNet as the image encoder. The user encoders still keep a similar fashion as the traditional IDRec architectures, e.g., SASRec [21], BERT4Rec [40] and NextItNet [52, 54].

A key advantage of MoRec models is that they are naturally transferable because item modality representation is universal regardless of platforms and systems. For example, ZESRec [7] proposed a zero-shot predictor by leveraging the natural language representation extracted from BERT. Similar work also includes UniSRec [18], IDA-SR [30], VQ-Rec [17] and ShopperBERT [38], which all leveraged textual features to realize transferable recommendation. However, so far, existing TransRec literature (especially image TransRec) mostly utilizes the off-the-shelf features pre-extracted from ME, which has efficiency advantages over fine-tuning heavy ME. Recently, [26, 39, 43, 55] started to perform joint training of user

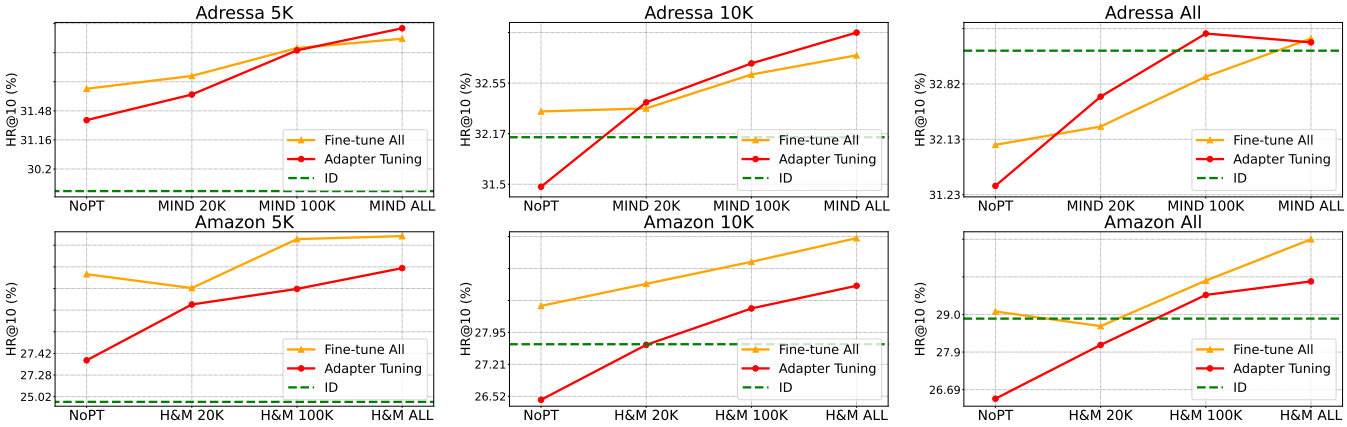


Figure 6: Scaling effects of fine-tuning and adapter-based TransRec using the SASRec objective. ID represents the ID-based recommender model (IDRec) with the same network architecture and training approach. The x-axis represents the size of pre-trained data. NoPT refers to TransRec that was not pre-trained by the source domain dataset.

encoder and item ME in both pre-training and fine-tuning stages (i.e., our FTA baseline), which showed significantly improved performance compared to the pre-extracted fixed features. Therefore, in this paper, we study TransRec as a comparison baseline in a more powerful end-to-end (or joint) learning manner.[10]

To the best of our knowledge, few studies have investigated the adapter tuning techniques for modality-based TransRec, especially for inserting adapters into item ME. PeterRec [51, 54] proposed the first adapter tuning technique for the recommendation task, but it highly relies on overlapped userIDs when performing transfer learning. Moreover, the majority of parameters in PeterRec are on the ID embedding layer rather than the middle layers.

9 CONCLUSION AND FUTURE WORK

In this paper, we conducted an extensive empirical study examining the performance of the popular Adapter Tuning (AdaT) techniques for modality-based TransRec models. We identified two facts: (1) the SOTA AdaT achieves competitive results compared to fine-tuning all parameters (FTA) for text recommendation; (2) AdaT falls slightly behind FTA for image recommendations in the cross-platform scenario but performs comparably in cross-domain scenarios. Then, we benchmarked four well-known AdaT approaches and found that their behavior was somewhat unique compared to NLP and CV tasks. We deeply studied several key factors that may influence AdaT results for recommendation tasks. At last, we found that TransRec with AdaT meets our expectations due to the ideal data scaling effect — TransRec benefits when upscaling the source domain data or downscaling the target domain data. Our work provides important guidelines for parameter-efficient transfer learning for modality recommendation models. It also has important practical implications for foundation models [1] in the RS community, with the grand goal of 'one model for all' [37, 43, 53].

There are several interesting future directions. The first one is to develop more advanced AdaT TransRec for cross-platform image recommendation. Then, we are also interested in investigating the effects of AdaT for multimodal (i.e., both text and image) TransRec. Third, given that most typical AdaT does not help to speed up the training process in practice (including for NLP and CV tasks), it is important to explore effective optimization techniques to reduce the computational cost and time for TransRec through end-to-end training of item modality encoders.

[10] Some recent work also performed end-to-end training for modality recommendation, but they [9, 46, 47] did not study the TransRec problem. We do not discuss them here.

REFERENCES

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[2] Tom Brown et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An efficient adaptive transfer neural network for social-aware recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 225–234.

[4] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. 2022. Vision Transformer Adapter for Dense Predictions. https://doi.org/10.48550/ARXIV.2205.08534

[5] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. https://doi.org/10.48550/ARXIV.2205.08084

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/ARXIV.1810.04805

[7] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318* (2021).

[8] Alexey Dosovitskiy, Lucas Beyer, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Shereen Elsayed, Lukas Brinkmeyer, and Lars Schmidt-Thieme. 2023. End-to-End Image-Based Fashion Recommendation. In *Recommender Systems in Fashion and Retail: Proceedings of the Fourth Workshop at the Recommender Systems Conference (2022)*. Springer, 109–119.

[10] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. 2019. Cross-domain recommendation without sharing user-relevant data. In *The world wide web conference*. 491–502.

[11] 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). *arXiv preprint arXiv:2203.13366* (2022).

[12] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The adressa dataset for news recommendation. In *Proceedings of the international conference on web intelligence*. 1042–1048.

[13] Wenjuan Han, Bo Pang, and Yingnian Wu. 2021. Robust Transfer Learning with Pretrained Language Models through Adapters. https://doi.org/10.48550/ARXIV.2108.02340

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16000–16009.

[15] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. 2022. Parameter-efficient Fine-tuning for Vision Transformers. https://doi.org/10.48550/ARXIV.2203.16329

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[17] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2022. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. *arXiv preprint arXiv:2210.12316* (2022).

[18] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 585–593.

[19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[20] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. 2022. Visual Prompt Tuning. https://doi.org/10.48550/ARXIV.2203.12119

[21] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. 197–206. https://doi.org/10.1109/ICDM.2018.00035

[22] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems* 34 (2021), 1022–1035.

[23] Walid Krichene and Steffen Rendle. 2020. On Sampled Metrics for Item Recommendation. In *KDD*.

[24] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691* (2021).

[25] Lei Li, Yongfeng Zhang, and Li Chen. 2023. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems* 41, 4 (2023), 1–26.

[26] Ruyu Li, Wenhao Deng, Yu Cheng, Zheng Yuan, Jiaqi Zhang, and Fajie Yuan. 2023. Exploring the Upper Limits of Text-Based Collaborative Filtering Using Large Language Models: Discoveries and Insights. *arXiv preprint arXiv:2305.11700* (2023).

[27] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/ARXIV.1907.11692

[28] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 1930–1939.

[29] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[30] Shanlei Mu, Yupeng Hou, Wayne Xin Zhao, Yaliang Li, and Bolin Ding. 2022. ID-Agnostic User Behavior Pre-training for Sequential Recommendation. https://doi.org/10.48550/ARXIV.2206.02323

[31] Weike Pan, Evan Xiang, Nathan Liu, and Qiang Yang. 2010. Transfer learning in collaborative filtering for sparsity reduction. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 24. 230–235.

[32] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterhub: A framework for adapting transformers. *arXiv preprint arXiv:2007.07779* (2020).

[33] Can Qin, Sungchul Kim, Handong Zhao, Tong Yu, Ryan A Rossi, and Yun Fu. 2022. External Knowledge Infusion for Tabular Pre-training Models with Dual-adapters. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1401–1409.

[34] Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476* (2023).

[35] Alec Radford, Jong Wook Kim, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

[36] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. In *https://shashankrajput.github.io/Generative.pdf*.

[37] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. 2021. 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*. 4104–4113.

[38] Kyuyong Shin, Hanock Kwak, Kyung-Min Kim, Minkyu Kim, Young-Jin Park, Jisu Jeong, and Seungjae Jung. 2021. One4all user representation for recommender systems in e-commerce. *arXiv preprint arXiv:2106.00573* (2021).

[39] Kyuyong Shin, Hanock Kwak, Kyung-Min Kim, Su Young Kim, and Max Nihlen Ramstrom. 2021. Scaling Law for Recommendation Models: Towards General-purpose User Representations. *arXiv preprint arXiv:2111.11294* (2021).

[40] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv e-prints* (2019), arXiv–1904.

[41] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. 2022. VL-Adapter: Parameter-Efficient Transfer Learning for Vision-and-Language Tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5227–5237.

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

[43] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. 2022. TransRec: Learning Transferable Recommendation from Mixture-of-Modality Feedback. *arXiv preprint arXiv:2206.06190* (2022).

[44] Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2020. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. *CoRR* abs/2002.01808 (2020). arXiv:2002.01808 https://arxiv.org/abs/2002.01808

[45] Xiaolei Wang, Kun Zhou, Ji-Rong Wen, and Wayne Xin Zhao. 2022. Towards Unified Conversational Recommender Systems via Knowledge-Enhanced Prompt Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1929–1937.

[46] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1652–1656.

[47] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. End-to-end Learnable Diversity-aware News Recommendation. *arXiv preprint arXiv:2204.00539* (2022).

[48] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.

[49] Yiqing Wu, Ruobing Xie, Yongchun Zhu, Fuzhen Zhuang, Ao Xiang, Xu Zhang, Leyu Lin, and Qing He. 2022. Selective fairness in recommendation via prompts. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2657–2662.

[50] Yiqing Wu, Ruobing Xie, Yongchun Zhu, Fuzhen Zhuang, Xu Zhang, Leyu Lin, and Qing He. 2022. Personalized Prompts for Sequential Recommendation. *arXiv preprint arXiv:2205.09666* (2022).

[51] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. 2020. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. *arXiv e-prints* (2020), arXiv–2001.

[52] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 582–590.

[53] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Joemon Jose, Beibei Kong, and Yudong Li. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 696–705.

[54] Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, et al. 2022. Tenrec: A Large-scale Multipurpose Benchmark Dataset for Recommender Systems. *arXiv preprint arXiv:2210.10629* (2022).

[55] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835* (2023).

[56] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, and Li Li. 2023. Denoising and Prompt-Tuning for Multi-Behavior Recommendation. *arXiv preprint arXiv:2302.05862* (2023).

[57] Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. 2021. Serial or parallel? plug-able adapter for multilingual machine translation. (2021).