User-Specific Adaptive Fine-Tuning for Cross-Domain Recommendations

Lei Chen, Fajie Yuan, Jiaxi Yang, Xiangnan He, Chengming Li, and Min Yang

Abstract—Making accurate recommendations for cold-start users has been a longstanding and critical challenge for recommender systems (RS). Cross-domain recommendations (CDR) offer a solution to tackle such a cold-start problem when there is no sufficient data for the users who have rarely used the system. An effective approach in CDR is to leverage the knowledge (e.g., user representations) learned from a related but different domain and transfer it to the target domain. Fine-tuning works as an effective transfer learning technique for this objective, which adapts the parameters of a pre-trained model from the source domain to the target domain. However, current methods are mainly based on the global fine-tuning strategy: the decision of which layers of the pre-trained model to freeze or fine-tune is taken for all users in the target domain. In this paper, we argue that users in RS are personalized and should have their own fine-tuning policies for better preference transfer learning. As such, we propose a novel User-specific Adaptive Fine-tuning method (UAF), selecting which layers of the pre-trained network to fine-tune, on a per user basis. Specifically, we devise a policy network with three alternative strategies to automatically decide which layers to be fine-tuned and which layers to have their parameters frozen for each user. Extensive experiments show that the proposed UAF exhibits significantly better and more robust performance for user cold-start recommendation.

Index Terms—Cross-domain recommendations, transfer learning, user-specific adaptive fine-tuning

1 Introduction

The past decade has seen a remarkable progress in deep learning (DL) and their applications in recommender systems (RS). A variety of neural network models [1], [2], [3], [4], [5] with larger and deeper architectures are proposed to model user interaction behaviors from online systems. Among them, sequential recommendation models, such as the GRU4Rec [1], NextItNet [2] and SASRec [4] have become especially popular since they in general require neither much feature engineering nor explicit user embeddings when making recommendations. Despite the success, deep neural network models tend to fail in practice when their training data (i.e., user interactions) is insufficient. Such scenarios widely exist in practical RS, when a large number of new users enroll in but have fewer interactions. Fortunately, the interaction behaviors of cold-start users are likely to be accessible from other online systems. For example, a user in Amazon who has few purchase records might have hundreds of clicking interactions in YouTube. Such observed interaction feedback by YouTube could be a clue to infer her preference and make recommendations in Amazon.[1] To this end, cross-domain recommendations (CDR) that transfer knowledge from a related source domain, have been proposed and become a popular way to tackle the recommendation problem of cold users.

Transfer learning (TL) based on pre-training and fine-tuning [6], [7], [8], [9] is widely employed for domain adaptation tasks. Its basic idea is to first pre-train a large neural network model with plenty of source data and then fine-tune it in the target domain where there might be insufficient training data. Fine-tuning has become a core technique for successful TL and been widely studied in computer vision (CV) [10] and natural language processing (NLP) [8]; however, thus far, it attracted relatively little attention in the recommendation field. Most of the previous work [11], [12], [13] used multi-task learning paradigm to learn the source domain and target domain tasks at the same time, which could not effectively transfer the knowledge from the source domain to the target domain, and would be easily overfitting when training examples are scarce in the target domain, compared with the pre-training and fine-tuning paradigm. Recent studies in PeterRec [14] and Conure [15] have took important steps towards universal user representation learning, showing that the two-

- *Lei Chen is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, P.R. China, and also with the University of the Chinese Academy of Sciences, Beijing 100049, P.R. China. E-mail: lei.chen@siat.ac.cn.*
- *Fajie Yuan is with Westlake University, Hangzhou 310024, P.R. China. E-mail: yuanfajie@westlake.edu.cn.*
- *Jiaxi Yang and Min Yang are with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, P.R. China. E-mail: {jx.yang, min.yang}@siat.ac.cn.*
- *Xiangnan He is with the University of Science and Technology of China, Hefei 230026, P.R. China. E-mail: xiangnanhe@gmail.com.*
- *Chengming Li is with the School of Intelligent Systems Engineering, Sun Yat-Sen University, Shenzhen 510275, P.R. China. E-mail: lichengming@mail.sysu.edu.cn.*

Manuscript received 14 April 2021; revised 22 September 2021; accepted 1 October 2021. Date of publication 14 October 2021; date of current version 3 February 2023.
Min Yang was partially supported by the National Natural Science Foundation of China under Grant 61906185, Youth Innovation Promotion Association of CAS China under Grant 2020357, Shenzhen Science and Technology Innovation Program under Grant KQTD20190929172835662, Shenzhen Basic Research Foundation under Grant JCYJ20200109113441941.
(Corresponding authors: Chengming Li and Min Yang.)
Recommended for acceptance by M. Wang.
Digital Object Identifier no. 10.1109/TKDE.2021.3119619

1. In this paper, we assume data of both domains is available and put aside privacy concerns.

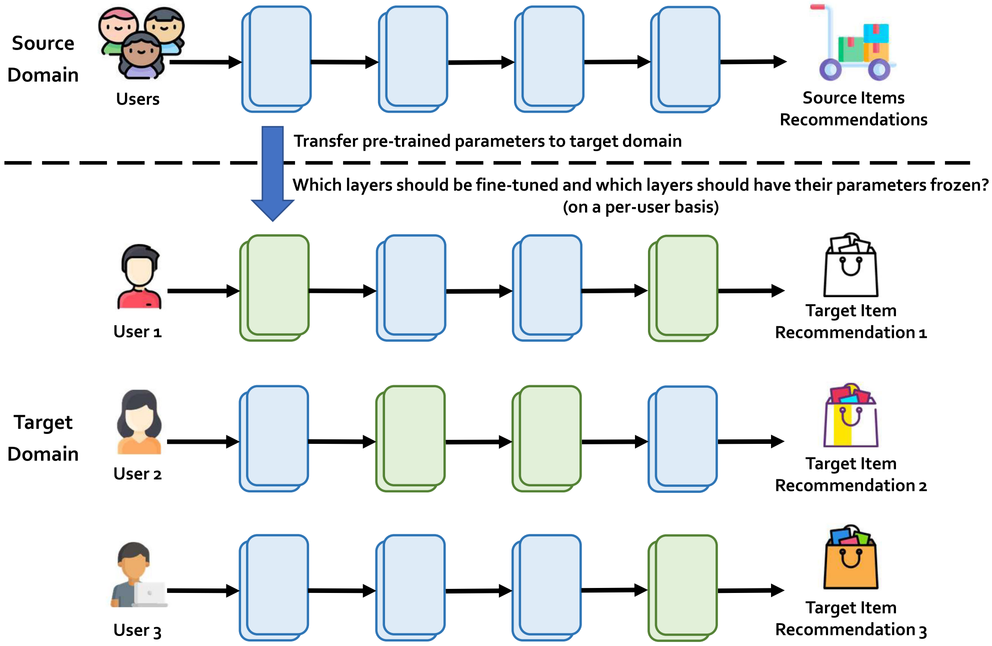


Fig. 1. Fine-tuning layers on a per-user basis. The blue color represents the pre-trained layers, while the green color denotes the fine-tuned layers.

stage TF paradigm significantly improves the accuracy of various downstream tasks by fine-tuning on the pre-trained model relative to training from scratch. Inspired by these works, in this paper we aim to investigate how to perform 'personalized' fine-tuning for cross-domain recommendations and encourage practitioners to apply our proposed fine-tuning as a common technique to further improve the performance of their CDR models.

There are several challenges when realizing the idea of fine-tuning of deep neural networks for CDR. (i) Great strides in accuracy have often been accompanied by increasing the depth and complexity of neural networks. Fine-tuning such large models is prone to overfitting when the target domain has insufficient training data, and when all parameters are optimized. (ii) Existing fine-tuning methods do not differentiate users — i.e., treating the pre-trained parameters of each user equally in the target domain. We argue that these methods restrict the power of the pre-trained representations, because some users in the target domain may have high similarity with the source domain, s.t. routing these users through pre-trained parameters might be a better choice compared to updating them with limited target data. Ideally, for each individual user, we expect the TL model is able to decide whether to use the pre-trained parameters or to fine-tune them, as illustrated in Fig. 1. For example, *User 1* requires the first and last layers to be fine-tuned, while the the two intermediate layers are kept the same as their pre-trained model. On the other hand, *User 3* requires more pre-trained parameters than *User 1* and *User 2* for the optimal recommendation accuracy in the target domain.

Motivated by the above-mentioned issues, in this paper we propose UAF, a user-specific adaptive fine-tuning method which tailors fine-tuning for effective CDR. To begin with, we first train a DL-based sequential recommendation model on a large source dataset and use it as our pre-trained model. This practice has been well verified by recent studies in [14], [15] (as well as in the NLP domains [16], [17], [18]) since sequential neural network can be trained in a self-supervised manner and thus could generate more universal user representations for effective transfer learning. In this paper, we instantiate the temporal convolutional neural network (CNN) model NextItNet [2] as the pre-trained backbone model given its efficient network structure [19], [20], superior performance [2], [21], [22] and widespread usage in modeling sequential recommendation data [14], [15], [22], [23], [24], [25] in recent literature. Specifically, our proposed UAF is a generic framework which can be directly applied to different sequential recommendation models like SASRec [4] and BERT4Rec [26]. Then, we fine-tune the pre-trained model on the target dataset using UAF to address the cold user problem. To be specific, for each user, we learn a policy network that makes binary decisions on whether to update or freeze the parameters of each layer in the pre-trained backbone network.

The main contributions of this paper are as follows:

- We propose a User-specific Adaptive Fine-tuning method (UAF) for CDR on cold-start users. UAF allows each user in the target domain to have their own fine-tuning policy. To the best of our knowledge, UAF is the first 'personalized' fine-tuning method designed for recommender systems.
- We propose a Gumbel-Softmax method (UAF-Gumbel), a reinforcement learning (RL) method (UAF-RL) and a soft method (UAF-Soft) to derive the optimal fine-tuning policy without suffering from the non-differentiable problem of the discrete policy decision functions.
- We instantiate UAF by using NextItNet [2] as the pre-trained backbone network, and report important results for both NextItNet and self-attention based network (i.e., SASRec [4]).

- Experimental results on five cross-domain datasets show that the proposed method achieves impressive results, especially when the training data in the target domain is insufficient.

2 Related Work

We recapitulate important work for cross-domain recommendations. Since our pre-trained model is based on sequential recommendation model, we also briefly review some representative work based on deep learning (DL).

2.1 Sequential Recommender Systems (SRS)

In general, DL-based sequential recommendation (SR) models can be classified into three categories, namely recurrent neural network (RNN) based [1], [27], convolutional neural network (CNN) based [2], [22], [28] and self-attention [4], [26] based methods. Specifically, Hidasi *et al.* [1] proposed the first DL-based SR model GRU4Rec by adapting RNN from language model in the NLP domain. Following this work, many extended RNN variants were proposed, which either optimized a new ranking loss [29], incorporated more context features [30], or developed more advanced data augmentation [27]. While effective, these models rely heavily on the hidden states of the entire past, which cannot take full advantage of the parallel processing resources (e.g., GPU and TPU) [2] during training. Therefore, CNN and self-attention based models are proposed to mitigate such limitations, becoming more popular in recent literature [2], [4], [26], [28]. Among them, Tang *et al.* [28] proposed Caser, which embeds a sequence of user-item interactions into an "image" and learn sequential patterns as local features of the image by using wide convolutional filters. Subsequently, Yuan *et al.* [2] proposed NextItNet, a deep temporal CNN-based recommendation model which particularly excels at modeling long-range item sequences. Later, GRec [22] improved NextItNet by considering two directional context information. CpRec [23] largely compressed NextItNet without hurting its recommendation accuracy. StackRec [21] and SkipRec [25] showed that NextItNet-style SR models could be stacked up to 100 layers for achieving its optimal accuracy, which distinguished from exiting work that usually applied less than 10 layers for evaluation. They also presented several methodologies to accelerate the training and inference processes of NextItNet. More recently, PeterRec [14] and Conure [15] demonstrated that the learned user representations by NextItNet were generic and could be transferred to solve various downstream recommendation problems. Meanwhile, self-attention based models, such as SASRec [4] and BERT4Rec [26], also showed competitive performance for the SRS task. In this paper, we plan to instantiate the well-known temporal CNN model (i.e., NextItNet) as the backbone network, given that the performance of which has been well evaluated by PeterRec [14] and Conure [15] for the CDR tasks [31]. Specifically, our proposed UAF is a generic framework which can be directly applied to different sequential recommendation models like SASRec [4] and BERT4Rec [26].

2.2 Cross-Domain Recommendations (CDR)

While deep learning (DL) based models have shown impressive performance in providing personalized recommendations, they often suffer from the cold-start problem, when new users enroll in a system but have a few or no labeled training data [14]. To deal with the cold-user problem, cross-domain recommendations (CDR) were proposed and proved effective. CDR transfer user preference between domains based on similarity of users and items that occur in both domains [32], [33]. CDR are particularly useful for recommendations of users who are cold in a target domain but have rich interactions from a source domain. According to existing literature, CDR methods can be broadly divided into two types: joint learning based CDR, referred to as JLCDR, and two-stage (pre-training + fine-tuning) transfer learning based CDR, referred to as TLCDR.

In terms of JLCDR, [11] proposed a DL-based domain adaptation approach, domain separation network [34], that solved the recommendation of cold-start users by jointly learning a stacked denoising autoencoder. Similarly, [12] explored an autoencoder and adversarial training to extract abstract rating patterns that were shared for same users across domains. CoNet [13] is another representative cross-domain recommendation model using deep neural networks as the base model. To enable effective knowledge transfer, CoNet introduced cross connections from the source network to the target and jointly trained objective functions of them. In addition, DDTCDR [35] developed a latent orthogonal mapping method to extract user preference over multiple domains while preserving connection between users across different latent spaces based on dual learning. [36] proposed CGN, which employs two generators to construct the dual-direction personalized itemset mapping between a user's behaviors in two different domains over time. CATN [37] proposed to model user preference transfer at the aspect-level derived from reviews, which does not require overlapping users or items in all domains. [38] proposed a mixed information flow network (MIFN) for cross-domain sequential recommendation to consider both the flow of behavioral information and the flow of knowledge by incorporating a behavior transfer unit and a knowledge transfer unit.

Despite its effectiveness, JLCDR approaches are not efficient during training since the joint learning (e.g., multi-task learning) scheme is usually very costly. In particular, to optimize the objective of the source domain is computationally very expensive as training data of it is often at a large-scale [8]. By contrast, TLCDR does not need to train the learner from scratch for target task. Besides, compared with JLCDR, TLCDR can obtain better prediction accuracy [15]. This is because the joint learning scheme of JLCDR that trades off many objective functions usually does not guarantee the optimal performance for the target objective function [14].

However, existing literature of TLCDR concentrated mainly on pre-training, i.e., developing a more expressive or efficient base model [8], [39], [40]. Closely related to this paper, a recent work called PeterRec [14] first evidenced that the pre-training model based on self-supervised learning on user sequential behaviors largely improved the fine-tuning accuracy for the target task (i.e., CDR in our case).

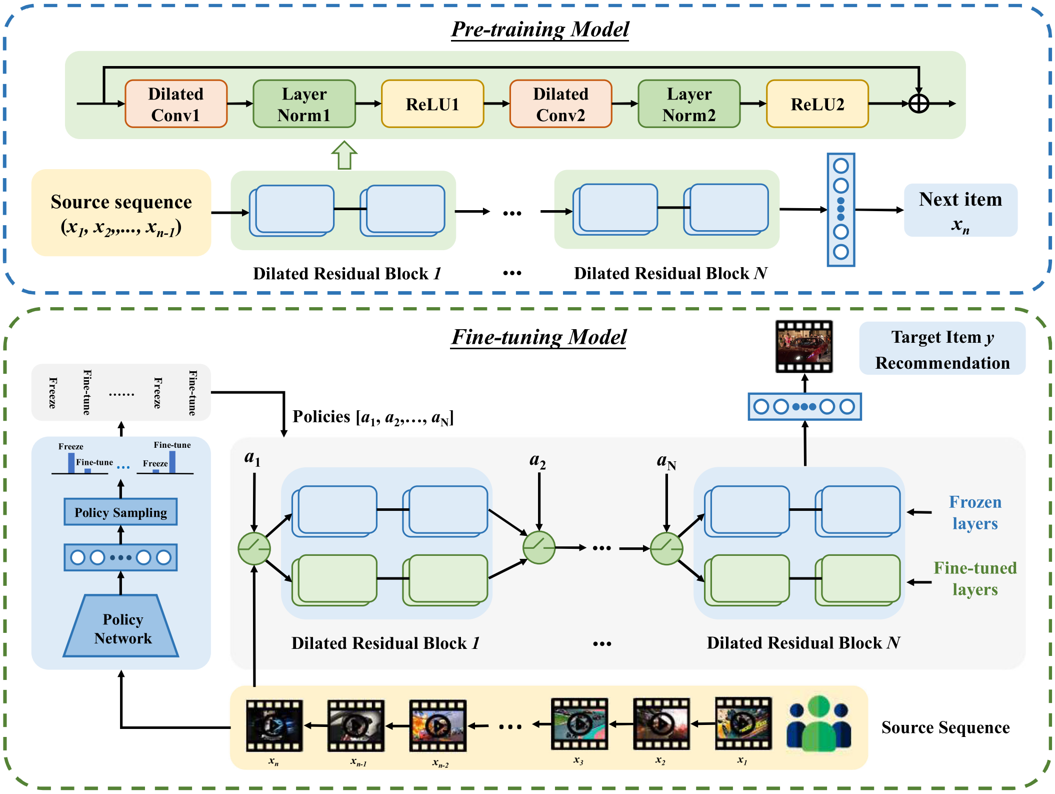


Fig. 2. Model architecture. The proposed UAF consists of two modules: pre-training (above) and fine-tuning (below). In the pre-trained model, we learn the backbone network (instantiated as NextItNet) on the source dataset to learn general user representations. In the fine-tuned model, we learn a user-specific adaptive fine-tuning strategy to automatically select reusable parameters of the pre-trained model on a per-user basis through a policy network.

Inspired by meta learning [41], PeterRec injected a small-scale adapter network into the pre-trained model and optimized only the adapter during fine-tuning. The authors showed that PeterRec performed significantly better than fine-tuning the entire model when training data in the target domain is scarce. In this paper, we argue that both PeterRec and the typical fine-tuning strategies achieve only sub-optimal accuracy since they perform fine-tuning globally for all users, ignoring the individual difference of users in target tasks. SpotTune [42] adopts dynamic routine in blocks in ResNet and improves the classification accuracy in CV area, which motivates us to explore personalized fine-tuning strategy in CDR. To this end, we present an adaptive fine-tuning neural network architecture on a per-user basis. In other words, our proposed fine-tuning method is personalized — each user is assigned to a different fine-tuning policy. To the best of our knowledge, UAF is the first attempt to the CDR task using a personalized fine-tuning strategy.

This paper can be regarded as an extension of our previous conference paper published in AAAI 2021 [25]. In [25], we propose a user-adaptive layer selection framework (SkipRec) by learning to skip inactive hidden layers on a per-user basis to address the inference problem for traditional sequential recommender systems with super deep layers. In this paper, we target at cross-domain recommendations, which is a completely different research topic. We propose a user-specific adaptive fine-tuning method (UAF) which tailors fine-tuning for effective cross-domain recommendations. In addition, we add a new UAF-Soft method by employing the weighted combination of freezed pre-trained layers and fine-tuned layers, which can preserve pre-trained information and receive valid fine-tuned information simultaneously.

3 Methodologies

3.1 Problem Definition

Suppose that there are two domains: a source domain $\mathcal{S}$ with a large number of user interactions and a target domain $\mathcal{T}$ with limited user interactions (i.e., the cold-user scenario). The input data takes the form of implicit feedback such as click logs and watching records. Let $\mathcal{U}$ be the set of users shared in both domains. Each instance in $\mathcal{S}$ contains a userID $u \in \mathcal{U}$, and the user interaction sequence $x^u = \{x_1^u, \ldots, x_n^u\} \in \mathcal{X}$, where x_t^u denotes the tth interaction of u and $\mathcal{X}$ (of size $|\mathcal{X}|$) is the set of items in source domain $\mathcal{S}$. Similarly, each instance in $\mathcal{T}$ consists of a userID u and item (label) $y \in \mathcal{Y}$, i.e., $(u, y) \in \mathcal{T}$, where $\mathcal{Y}$ (of size $|\mathcal{Y}|$) is the set of items in target domain $\mathcal{T}$.

Our objective is to recommend a set of items (a.k.a, the top-N item recommendation) in the target domain to the user u who has limited interactions but possesses sufficient sequential behaviors in the source domain.

3.2 The Overall Architecture

Fig. 2 illustrates the overall architecture of the proposed UAF method. The training procedure of UAF consists of two stages: pre-training and fine-tuning. In the first stage, we pre-train the backbone network (instantiated as NextItNet) on the source dataset with numerous user-item interactions. The pre-trained model is supposed to learn general user representations which are reusable for same users in the target dataset. Then, we propose a user-specific adaptive fine-tuning strategy to automatically select reusable layers from the pre-trained model on a per-user basis.

3.2.1 Pre-Training Model

As mentioned before, we instantiate NextItNet as the pre-trained model to learn the user representations given its superior performance in recent literature [2], [21]. Specifically, our proposed UAF is a generic framework which can be directly applied to different sequential recommendation models like SASRec [4] and BERT4Rec [26]. NextItNet is composed of a stack of dilated convolutional layers, every two of which are wrapped by a residual block structure. To be specific, each input item x_i^u is converted into an embedding vector $\mathbf{e}_i^u$, and the item sequence x^u can be represented as an embedding matrix $\mathbf{E}^u = [\mathbf{e}_1^u \dots \mathbf{e}_n^u]$. Then, we pass the item embeddings into a stack of dilated convolutional (DC) layers to learn the feature vector $\mathbf{E}_l^u$ which is expected to capture sequential dependencies. Formally, the lth residual block with the DC operation is formalized as

$$\mathbf{E}_l^u = \mathcal{F}_l(\mathbf{E}_{l-1}^u) + \mathbf{E}_{l-1}^u, \tag{1}$$

where $\mathbf{E}_{l-1}^u \in \mathbb{R}^{n \times k}$ and $\mathbf{E}_l^u \in \mathbb{R}^{n \times k}$ are the input and output of the lth residual block considered, k is the item embedding size. $\mathcal{F}_l(\mathbf{E}_{l-1}^u) + \mathbf{E}_{l-1}^u$ is a shortcut connection by element-wise addition. $\mathcal{F}_l(\mathbf{E}_{l-1}^u)$ represents the residual mappings as

$$\mathcal{F}_l(\mathbf{E}_{l-1}^u) = \sigma\big(\mathbf{LN}_2\big(\psi_2\big(\sigma\big(\mathbf{LN}_1\big(\psi_1(\mathbf{E}_{l-1}^u)\big)\big)\big)\big)\big), \tag{2}$$

where ψ_1 and ψ_2 represent the casual convolution operations, which makes sure the model cannot violate the ordering in modeling the data, that is, the prediction by the model cannot depend on any of the future timesteps. $\mathbf{LN_1}$ and $\mathbf{LN_2}$ represent layer normalization functions. σ is ReLU activation function. Finally, a softmax output layer is applied to predict the probability distribution for the next item x_{n+1}^u

$$p(x_{n+1}^u | x_{1:n}^u) = softmax(\mathbf{W}\mathbf{E}_l^u + \mathbf{b}), \tag{3}$$

where $\mathbf{W}$ is the projection matrix, and $\mathbf{b}$ is the bias term.

The joint probability $p(x^u; \Omega)$ of each user sequence is computed by the product of the conditional distributions over the interacted items as follows:

$$p(x^u; \Omega) = \prod_{i=1}^{n} p\big(x_i^u | x_{1:i-1}^u; \Omega\big), \tag{4}$$

where $p\big(x_i^u | x_{1:i-1}^u; \Omega\big)$ is the predicted probability for the ith item x_i^u conditioned on all its previous interactions $\{x_1^u, \dots, x_{i-1}^u\}$, and Ω is the set of parameters.

The objective function $\mathcal{G}(\mathcal{S}; \Omega)$ of pre-trained NextItNet is to minimize the sum of negative log-likelihood of the joint probability, which is defined as

$$\begin{aligned} \mathcal{G}(\mathcal{S}; \Omega) &= -\sum_{x^u \in \mathcal{S}} \log p(x^u; \Omega) \\ &= -\sum_{x^u \in \mathcal{S}} \log \prod_{i=1}^{n} p\big(x_i^u | x_1^u, \dots, x_{i-1}^u; \Omega\big). \end{aligned} \tag{5}$$

3.2.2 User-Specific Adaptive Fine-Tuning

After the pre-training process, we employ UAF to identify which layers in the pre-trained model are required to be fine-tuned or freezed. For the lth residual block in the pre-trained backbone model, in order to decide whether or not to fine-tune it during training, we first freeze the original block $\mathcal{F}_l$ and then create a new trainable block $\hat{\mathcal{F}}_l$, which is initialized with the parameters of $\mathcal{F}_l$. With the additional block $\hat{\mathcal{F}}_l$, the output of the lth residual block in UAF is computed as

$$\mathbf{E}_l^u = \mathbf{I}_l(\mathbf{E}^u)\hat{\mathcal{F}}_l\big(\mathbf{E}_{l-1}^u\big) + (1 - \mathbf{I}_l(\mathbf{E}^u))\mathcal{F}_l\big(\mathbf{E}_{l-1}^u\big) + \mathbf{E}_{l-1}^u, \tag{6}$$

where $\mathbf{I}_l(\mathbf{E}^u)$ is a binary policy variable that indicates whether the residual block should be frozen or fine-tuned, conditioned on the input user sequence $\mathbf{E}^u$. During training, given an input user sequence, the frozen block $\mathcal{F}_l$ trained on the source domain dataset is left unchanged, and the replicated block $\hat{\mathcal{F}}_l$ can be optimized towards the target domain dataset. Therefore, the input user sequence can either share the frozen block $\mathcal{F}_l$ or fine-tune the residual block $\hat{\mathcal{F}}_l$. $\mathbf{I}_l(\mathbf{E}^u)$ is sampled from a discrete distribution with two categories (freeze or fine-tune), which is parameterized by the output of a lightweight policy network. To be more specific, if $\mathbf{I}_l(\mathbf{E}^u) = 0$, then the lth frozen residual block is reused; otherwise, the lth replicated residual block is fine-tuned.

By performing a series of convolution operations on frozen and fine-tuned layers with the input user sequence $\mathbf{E}^u$, we obtain the final hidden vector $\mathbf{h}_n$ of the last layer. Following [14], a fully-connected layer is applied to predict the score of item y in the target domain by:

$$\mathbf{o}_y = \mathbf{W}_n \mathbf{h}_n + \mathbf{b}_n \tag{7}$$

where $\mathbf{W}_n$ and $\mathbf{b}_n$ are parameters to be learned.

During the fine-tuninng stage, we adopt the popular pairwise ranking loss (BPR) [43] as the objective function $\mathcal{L}_{BPR}(\mathcal{T}; \Theta)$ of UAF for top-N recommendation

$$\mathcal{L}_{BPR}(\mathcal{T}; \Theta) = -\sum_{(u,y) \in \mathcal{T}} \log\big(\delta\big(\mathbf{o}_{y^+} - \mathbf{o}_{y^-}\big)\big), \tag{8}$$

where Θ is the parameters to be optimized, including pre-trained parameters (excluding the softmax matrix) and fine-tuned parameters (new softmax layer and newly added residual blocks), δ is the logistic sigmoid function, y^+ is the positive label and y^- is a negative label randomly sampled from $Y \backslash y^+$ following [43], $\mathcal{T}$ contains all positive pairs (u, y^+) and the selected negative pairs (u, y^-) in the target dataset.

3.3 Policy Network

To derive the optimal fine-tuning strategy given the input user sequence, we develop a policy network to output a binary policy vector, representing the actions to freeze or fine-tune each residual block in the pre-trained backbone network. The policy network is instantiated using a lightweight NextItNet model with dilations of {1, 2, 4, 8} (4 layers or 2 residual blocks). Without any restrictions, the policy network can also be implemented with other deep neural networks, e.g., a RNN model. In this paper, we propose three alternative strategies (i.e., UAF-Gumbel, UAF-RL and UAF-Soft) to optimize the policy network.

3.3.1 UAF-Gumbel Method

Since the policy $\mathbf{I}_l(\mathbf{E}^u)$ is a discrete binary variable, it is intractable to optimize the policy network with backpropagation due to the non-differentiable problem. To resolve this issue, we propose using the Gumbeling Softmax sampling method [44], [45] to generate the actions (freeze or fine-tune) from a discrete distribution. We refer UAF with Gumbeling Softmax training as *UAF-Gumbel*.

UAF-Gumbel draws samples $\mathbf{z}$ from a categorical distribution with class probabilities $\{\pi_1, \pi_2, \ldots, \pi_k\}$. Here, we have $k = 2$, indicating the freezing or fine-tuning actions. That is, for each residual block, π_1 and π_2 are scalars corresponding to the final output of the policy network

$$\mathbf{z} = \text{ one\_hot } \left(\arg\max_i [g_i + \log \pi_i]\right), \tag{9}$$

where $\mathbf{z}$ is a one-hot vector and $\{g_1, g_2, \ldots, g_k\}$ are i.i.d samples drawn from $Gumbel(0,1)$ distribution. The $Gumbel(0,1)$ distribution can be sampled using inverse transform sampling by drawing u from a uniform distribution, i.e., $u \sim Uniform(0,1)$ and we can compute $g = -\log(-\log(u))$.

The $\arg\max$ operation in Eq. (9) is non-differentiable, but we can use the Gumbel Softmax distribution, which adopts softmax as a continuous relaxation to $\arg\max$ in order to allievate the non-differentiable problem. We relax the one-hot encoding of $\mathbf{z}$ to a real-valued vector $\boldsymbol{\alpha}$ using

$$\alpha_i = \frac{\exp((\log \pi_i + g_i)/\tau)}{\sum_{j=1}^{k} \exp\big((\log \pi_j + g_j)/\tau\big)} \qquad \text{for } i = 1, \ldots, k, \tag{10}$$

where τ is a temperature parameter to control the discreteness of the output vector $\boldsymbol{\alpha}$. When τ becomes closer to 0, the samples from the Gumbel Softmax distribution become indistinguishable from the discrete distribution, i.e., almost the same as the one-hot vector $\mathbf{z}$. Here we set τ to 10 by default.

Sampling the fine-tuned policy $\mathbf{I}_l(\mathbf{E}^u)$ from the Gumbel Softmax distribution allows us to backpropagate from the discrete binary decision samples to the policy network, as the Gumbel Softmax distribution is smooth for $\tau > 0$ and has well-defined gradients with respect to the parameters π_i. Similar to [42], [46], we generate all freezing/fine-tuning policies for all residual blocks at once for the trade-off of efficiency and accuracy. During the forward pass, we sample the fine-tuning policy $\mathbf{I}_l(\mathbf{E}^u)$ using Eq. (9) for the lth residual block. As for the backward pass, we approximate the gradients of the discrete samples by computing the gradients of the continuous softmax relaxation in Eq. (10).

By using the above approach, we can easily achieve UAF-Gumbel method in a differentiable way and obtain the policy regarding which layers in pre-trained model should be fine-tuned. By using the objective function in Eq. (8) for the target domain, the policy network is jointly trained with the pre-trained backbone model in an end-to-end way.

3.3.2 UAF-RL Method

In fact, we can learn the policy network by employing the reinforcement learning (RL) algorithm (called *UAF-RL*) which is a popular technique for solving the non-differentiable problem. The main idea is to learn the policy network that outputs the posterior probabilities of all the binary decisions for freezing or fine-tuning each block in the fine-tuning network. The policy network is trained using curriculum learning [47] to maximize a reward that incentivizes the use of as few blocks to fine-tune as possible while preserving the prediction accuracy. In this regard, we can consider the potential trade-offs between computational cost and prediction accuracy. We can model the UAF-RL method as a markov decision process (MDP). The *state* is represented by the user representation which is obtained by modeling the interaction sequence through the policy network. The *action* is represented by the fine-tuning strategy derived by the UAF-RL with the user representation. We design the *reward* function based on the following considerations: (i) enabling the policy network to control the fine-tuning network so as to generate well-recommended items in target domain; (ii) achieving a significant computational cost reduction to meet the requirement of online service. When taking the corresponding actions, we compute the rewards and optimize the entire model to maximize the expectations of the cumulative rewards.

Specifically, let $\mathcal{A} = \{0/1\}^N \in \mathbb{R}^N$ denote the fine-tuning policies predicted by the policy network, where N represents the number of residual blocks in the fine-tuning network. In particular, $\mathcal{A} \sim p(\pi_i)$, where $\pi_i \in \{\pi_1, \pi_2\}$, indicating the freezing or fine-tuning actions. In order to evaluate the advantage of an action $\mathcal{A}_l$, we define the reward function as the following formula:

$$\mathbf{R}(\mathcal{A}) = \begin{cases} 1 - \left(\sum_l \mathbb{1}(\mathcal{A}_l)/N\right)^2 & \text{if correct} \\ -\gamma & \text{otherwise} \end{cases}, \tag{11}$$

where $\mathbb{1}(\cdot)$ is an indicator function, $\mathcal{A}_l$ represents the freezing or fine-tuning policy applied on the lth block, γ is a hyperparameter to penalize the wrong policy. Specifically, $\left(\sum_l \mathbb{1}(\mathcal{A}_l)/N\right)^2$ measures the percentage of blocks that are fine-tuned. When a correct recommendation is produced, we incentivize block freezing by giving a larger reward to a policy that uses fewer blocks to fine-tune. In addition, we penalize incorrect recommendations with γ, which controls the trade-off between efficiency and effectiveness. γ is simply set to 1 in this paper.

To optimize the policy network, we adopt self-critical sequence training (SCST) [48], which is a form of the popular REINFORCE [49] algorithm, for model training. Rather than estimating a "baseline" to normalize the rewards and reduce variance, SCST applies the output of its own test-time inference algorithm to normalize the rewards it experiences. In details, the exploration action $\mathcal{A}_l^s$ is obtained by sampling from the categorical distribution $p(\pi_i)$ through modeling the source sequence, while the self-critical baseline is calculated by the greedy search, and the policy take action by $\hat{\mathcal{A}}_l = \arg\max_i p(\pi_i)$. To learn the optimal parameters of the policy network, we minimize the following SCST loss

$$\mathcal{L}_{RL} = -\sum_{l=1}^{N} \log p(\mathcal{A}_l^s)\Big(\mathbf{R}(\mathcal{A}_l^s) - \mathbf{R}(\hat{\mathcal{A}}_l)\Big), \tag{12}$$

where $p(\mathcal{A}_l^s)$ represents the probability to sample the exploration action $\mathcal{A}_l^s$. Since the self-critical baseline is based on the test-time estimate under the current model, SCST is forced to improve the performance of the model under the inference algorithm used at test time. At inference stage, we obtain the actual fine-tuning policy $\mathbf{I}_l(\mathbf{E}^u)$ by the greedy search $\arg\max_i p(\pi_i)$.

Finally, we jointly train the policy network and fine-tuning network in an end-to-end way and optimize the weighted-sum of the BPR loss and SCST loss in the UAF-RL method

$$\mathcal{L} = \mathcal{L}_{BPR} + \beta\mathcal{L}_{RL}, \tag{13}$$

where $\mathcal{L}_{BPR}$ and $\mathcal{L}_{RL}$ represent the standard BPR loss and the SCST loss, respectively, and β is a hyperparameter to control the weight of the SCST loss, which is set to 1 in our experiments.

3.3.3 UAF-Soft Method

In the UAF-Gumbel and UAF-RL methods, we learn a policy network to make pure binary decisions for fine-tuning or freezing the parameters of each layer in the pre-trained backbone model. That is, the decision $\mathbf{I}_l(\mathbf{E}^u)$ in every residual block is either 0 or 1. As an alternative, we propose a *UAF-Soft* method to learn the policy network by employing the weighted combination of freezed pre-trained layers and fine-tuned layers. In such a UAF-Soft method, the output of each residual block is no longer from a certain layer, but from both pre-trained layers and fine-tuned layers. In this way, we can preserve the useful pre-trained information and receive valid fine-tuned information simultaneously.

Specifically, we design a gate for each residual block to control the flow of parameters in the freezed pre-trained layers and fine-tuned layers in a soft way. The weight ratio in each gate is still obtained from the policy network using the lightweight NextItNet model with dilations of {1, 2, 4, 8}. We obtain the user representation by modeling the interaction sequence from source domain through the policy network, with passing into the residual block to obtain the final hidden vector $\mathbf{h}_p$. Correspondingly, we will get $\mathbf{v}$ for all residual blocks during the fine-tuning stage and adopt a logistic sigmoid function δ to achieve the weight distribution of the fine-tuning policy $\mathbf{I}_l(\mathbf{E}^u)$

$$\mathbf{v} = \mathbf{W}_p\mathbf{h}_p + \mathbf{b}_p \tag{14}$$

$$\mathbf{I}_l(\mathbf{E}^u) = \delta(\mathbf{v}), \tag{15}$$

where $\mathbf{W}_p$ is a projection matrix, and $\mathbf{b}_p$ is a bias term.

Following UAF-Gumbel and UAF-RL, we can obtain the output of the lth residual block $\mathbf{E}_l^u$ at the fine-tuning stage using Eq. (6). Since the sigmoid function is differentiable, we can directly train the policy network and the fine-tuning model in an end-to-end way.

4 Experimental Setup

4.1 Experimental Datasets

To evaluate the effectiveness of UAF, we collect four transfer learning datasets from Tencent.[2] Each of them is either from different recommender systems or has different properties. Among them, two datasets have been made publicly available in [14]. In addition, we conduct extensive experiments on the widely used MovieLens dataset to evaluate the performance of our proposed UAF. The detailed properties of these datasets are described as follows.

4.1.0.1 ColdRec-1. ColdRec-1 includes both source and target domains. The source domain is the news recommendation dataset collected from QQ Browser.[3] Each instance is formed of a sequence of n watching interactions of a user, where n is set to 50. Sequence length less than 50 will be padded with zero in the beginning, similar to [14]. The target dataset is collected from Kandian.[4] All users in Kandian are cold with at most 3 interactions (including clicks of news, videos or advertisements) and half of them have only one interaction. The source and target domains are connected by userID as all users in the target dataset also have corresponding watching records in the source dataset.

4.1.0.2 ColdRec-2. It has similar characteristics with ColdRec-1 except that the source dataset contains 100 watching interactions for each user in the sequence session. In addition, the source dataset includes both news and video interactions. The target dataset has at most 5 interactions for each user.

4.1.0.3 ColdRec-3. This is a private dataset collected by Tencent *Oula* team. Similar to ColdRec-1, the source domain is the news recommendation dataset of QQ Browser, and the session length is set to 50. The target domain is an advertisement recommendation dataset, where 90% users in target dataset have only one clicking action.

4.1.0.4 ColdRec-4. This is also collected by Tencent *Oula* team. The source dataset is collected from Tencent Video[5] including TV series, movies and short videos, and the session length is set to 50. The target dataset is collected from Kandian, and each user in target dataset has at most 5 clicks on news and videos.

4.1.0.5 MovieLens. MovieLens[6] is a widely used benchmark dataset in recommendation systems (RS). The version we used includes 25 million movie rating records over thousands of movies and users. Each instance in source domain contains a userID and his recent 30 clicking (excluding 4- and 5-star) interactions and each instance in target domain contains a userID and an item that is rated higher than 4, following [15].

The statistics are summarized in Table 1.

4.2 Baselines and Evaluation

We compare our models (UAF-Gumbel, UAF-RL and UAF-Soft) with several powerful approaches.

- *CoNet*: CoNet [13] is a widely used CDR baseline which transfers information between different domains by a cross-stitch network [50] and the information in each domain is captured by the neural collaborative filtering (CF) model [3].
- *DDTCDR*: DDTCDR [35] is another representative CDR model which develops a latent orthogonal

2. https://www.tencent.com/en-us/
3. https://browser.qq.com
4. https://sdi.3g.qq.com/v/2019111020060911550
5. https://v.qq.com/
6. https://www.grouplens.org/datasets/movielens/

TABLE 1
Statistics of the Five Experimental Datasets

Domain	ColdRec-1		ColdRec-2		ColdRec-3		ColdRec-4		MovieLens	
	Instances	Items	Instances	Items	Instances	Items	Instances	Items	Instances	Items
$\mathcal{S}$	1,649,095	191,022	1,472,428	645,981	22,360	11,349	340,858	52,171	677,259	49,720
$\mathcal{T}$	3,798,114	20,343	2,947,688	17,880	24,643	162	1,267,117	6,288	3,058,638	26,311

We reported the total number of instances and items in both source domain and target domain in each dataset. We regard the user sequence in source domain with each item in target domain of the same user as an instance of the target domain. Each user in the target dataset has several interactions, we form the training instances by using each interaction in the target dataset with the same user sequence in source domain. That is, if the target domain has 3 interactions, then we can generate 3 instances (maybe 2 for training and 1 for testing). We followed the same settings with previous work [14].

mapping method to extract user preference over multiple domains while preserving connection between users across different latent spaces based on dual learning.

- *Finetune-Zero*: It is only trained on the target dataset without performing pre-training on the source dataset. That is, its parameters are randomly initialized.
- *Finetune-CLS*: It is pre-trained on the source dataset. We merely fine-tune the softmax layer of the model on the target dataset.
- *Finetune-Last-N* (N =1, 2): We pre-train the model on the source dataset and fine-tune the last N residual blocks of the pre-trained network (with the softmax layer) on the target data following [51].
- *Finetune-All*: We pre-train the model on the source dataset and fine-tune all parameters on the target dataset. Finetune-All is the most widely adopted fine-tuning approach, and usually performs better than Finetune-CLS and Finetune-Last-N as the entire model is tuned [14].
- *MTL*: We present a widely used multi-task learning (referred to as MTL) baseline by hard parameter sharing [52], similar to that used in [15]. MTL jointly learns the objective functions of both source and target domains rather than using the two-stage pre-training and fine-tuning framework.
- *PeterRec*: This is a recently proposed transfer learning framework which can be used to solve the cold-user problem [14]. The basic idea is to fine-tune a small-sized adapter neural network inserted in the pre-trained network. We evaluate PeterRec by using the official code[7] under our setting.

Our evaluation follows [14] by adopting two popular top-N metrics to measure the quality of recommended items, including MRR@N (Mean Reciprocal Rank) [1] and HR@N (Hit Ratio) [53]. Here N is set to 5 for comparison.

4.3 Implementation Details

We randomly split each target dataset into training (80%), validation (5%) and test (15%) sets, similar to [14]. The grid search algorithm is applied on the validation sets to tune the hyperparameters. The size of each item embedding is set to be 128 for all the models. The NextItNet architecture is implemented by using 16 dilation layers or 8 residual blocks (i.e., {1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8, 1, 2, 4, 8}), which is used as the backbone of the pre-training and fine-tuning models. Specifically, our proposed UAF is a generic framework which can be directly applied to different sequential recommendation models like SASRec [4] and BERT4Rec [26]. We employ Adam optimizer to train our models with learning rate η of 0.0001. Batch size b and kernel size are set to be 256 and 3, respectively. Regarding the policy network, we use dilations of {1, 2, 4, 8} (4 layers or 2 residual blocks), which is a lightweight neural network. Note that, as mentioned before, without any restrictions, the policy network can also be implemented with other deep neural networks, e.g., a RNN model. All the experiments are implemented in TensorFlow and trained on a single TITAN RTX GPU.

7. https://github.com/fajieyuan/sigir2020_peterrec

5 Experimental Results

As the key contribution of this work is to solve the CDR task with cold (or new) users through a user-specific adaptive fine-tuning method UAF, we evaluate UAF on five real-world datasets and conduct extensive experiments to answer the following research questions:

(1) *RQ1:* Whether the proposed UAF (i.e., UAF-Gumbel, UAF-RL and UAF-Soft) perform better than the traditional CDR baselines and existing fine-tuning approaches?
(2) *RQ2:* Is UAF a general method or can it adapt to other sequential recommendation models directly?
(3) *RQ3:* What are the effects of the policy network? Does it enable adaptive fine-tuning for each user? Can our policy network be replaced by other types of neural networks (e.g., a RNN model)?
(4) *RQ4:* Can UAF stay robust when the training data in the target domain is insufficient?
(5) *RQ5:* Does UAF still work well in other tasks (e.g., user profile prediction)?

5.1 Performance Comparison (RQ1)

Table 2 reports the overall results of UAF and baselines on the five datasets. We can make the following observations. First, CoNet and DDTCDR which adopt multi-task learning paradigm achieve similar performance on all the five datasets with the MTL baseline and perform relatively poor compared to the fine-tuning methods. Second, Finetune-Zero performs terrible among all baselines on ColdRec-1/2/4 and MovieLens in terms of both MRR@5 and HR@5. In particularly, by comparing Finetune-Zero with Finetune-All on the three datasets, we can conclude that pre-training on the source domain is of great help to achieve better recommendation accuracy. On the other hand, Finetune-All

TABLE 2
Overall Results in Terms of MRR@5 and HR@5 on All the Five Datasets

Models	ColdRec-1		ColdRec-2		ColdRec-3		ColdRec-4		MovieLens	
	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5
CoNet	0.2268	0.3944	0.3589	0.5403	0.1117	0.2098	0.1976	0.3342	0.6586	0.8792
DDTCDR	0.2427	0.4119	0.3651	0.5496	0.1175	0.2190	0.2045	0.3451	0.6778	0.8924
Finetune-Zero	0.2332	0.4013	0.3632	0.5464	0.1174	0.2150	0.1929	0.3257	0.6050	0.8427
Finetune-CLS	0.2419	0.4124	0.3712	0.5547	0.0909	0.1657	0.1935	0.3283	0.6645	0.8862
Finetune-Last1	0.2546	0.4307	0.3864	0.5724	0.1062	0.1918	0.2032	0.3433	0.6738	0.8911
Finetune-Last2	0.2575	0.4337	0.3885	0.5753	0.1176	0.2118	0.2073	0.3485	0.6783	0.8933
Finetune-All	0.2565	0.4345	0.3880	0.5758	0.1185	0.2192	0.2077	0.3495	0.6910	0.8993
MTL	0.2410	0.4114	0.3633	0.5468	0.1169	0.2181	0.2093	0.3527	0.6783	0.8919
PeterRec	0.2578	0.4356	0.3886	0.5763	0.0993	0.1795	0.2117	0.3550	0.6884	0.8966
UAF-Gumbel	0.2594	0.4378	0.3940	0.5824	0.1190	**0.2212**	**0.2151**	**0.3640**	0.6942	0.9053
UAF-RL	0.2599	0.4386	**0.3942**	0.5837	0.1196	0.2197	0.2142	0.3619	**0.6959**	**0.9075**
UAF-Soft	**0.2600**	**0.4397**	0.3939	**0.5841**	**0.1209**	0.2189	0.2140	0.3613	0.6936	0.9041

Note that the improvements of UAF over all baseline models are statistically significant in terms of paired t-test with p-value < 0.01.

obtains only 1% improvement over Finetune-Zero on ColdRec-3. The result suggests that the correlation between the source and target domains on ColdRec-3 may be not as significant as ColdRec-1/2/4. The observation is reasonable since advertisement recommendation may have some discrepancy with news recommendation in terms of user preference. Third, it is shown that Finetune-Last2 performs better than Finetune-Last1, which further performs better than Finetune-CLS. Our observations here are consistent with those in prior work. Fourth, MTL performs better than Finetune-Zero except on ColdRec-3, indicating the effectiveness of joint learning. Furthermore, Finetune-All outperforms MTL on ColdRec-1/2/3 and Movielens but slightly underperforms it on ColdRec-4. We argue that the optimal parameters learned for two or more objectives in MTL does not guarantee the optimal performance on the objective of the target domain. By constrast, Finetune-All only cares about the objective on the target domain and thus performs generally better. Fifth, PeterRec achieves modest results on ColdRec-1/2/4 and Movielens, but shows worse results on ColdRec-3. The result further indicates that the five datasets have different characteristics in terms of knowledge transfer: some datasets benefit a lot from pre-training, while others may not. Again, pre-training on ColdRec-3 is not as important as on other datasets, and as a result, PeterRec exhibits much worse performance because all pre-trained parameters in PeterRec are not allowed to be modified.

Meanwhile, we observe that UAF, including UAF-Gumbel, UAF-RL and UAF-Soft, performs obviously the best on ColdRec-2/3/4 and Movielens and on par with PeterRec and Finetune-All on ColdRec-1. In fact, we further find that UAF performs significantly better than all other baselines even on ColdRec-1 with limited training data, as shown in Section 5.4. We believe the key advantage of UAF mainly comes from its adaptive fine-tuning mechanism since these models are based on the same backbone network. By accomplishing the goal of per user per fine-tuning policy, UAF achieves more intelligent transfer learning across domains.

5.2 Generality of the UAF (RQ2)

To verify the generality of our proposed UAF, we specify it with SASRec [4] and the results are reported in Table 3. Similar conclusions can be made with BERT4Rec [26]. Regarding the policy network, we implement it with a lightweight multi-head self-attention network, i.e., one standard attention residual block.

As shown, we can draw similar conclusions as discussed in Section 5.1. Our proposed UAF methods (including UAF-Gumbel-SASRec, UAF-RL-SASRec and UAF-Soft-SASRec) achieve notably better recommendation results than other baselines, which demonstrates the generality of UAF while adapting it to SASRec. The main benefits of UAF come from the attention policy, which is as powerful as the convolutional and recurrent networks.

5.3 Ablation Studies on the Policy Network (RQ3)

To answer RQ3, we explore the following ablation studies to verify the effects of the policy network in UAF. Note that we may only report partial results for clarity if similar behaviors are observed.

TABLE 3
Results by Generalizing UAF to SASRec in Terms of MRR@5 and HR@5 on All the Five Datasets

Models	ColdRec-1		ColdRec-2		ColdRec-3		ColdRec-4		MovieLens	
	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5
Finetune-Zero	0.1821	0.3262	0.3057	0.4725	0.0892	0.1576	0.1778	0.3015	0.4746	0.7192
Finetune-All	0.2492	0.4256	0.3758	0.5527	0.1237	0.2302	0.2111	0.3560	0.6668	0.8897
PeterRec	0.2578	0.4356	0.3886	0.5763	0.0993	0.1795	0.2117	0.3550	0.6884	0.8966
UAF-Gumbel-SASRec	0.2573	0.4374	0.3875	0.5781	0.1270	0.2345	0.2153	0.3638	0.6880	0.8953
UAF-RL-SASRec	**0.2591**	**0.4418**	0.3881	**0.5786**	**0.1308**	0.2351	0.2159	0.3644	**0.6898**	0.8985
UAF-Soft-SASRec	0.2584	0.4389	**0.3893**	0.5783	0.1297	**0.2368**	**0.2186**	**0.3684**	0.6893	**0.8999**

TABLE 4
Results With a Random Policy (Termed as UAF-Random) in Terms of MRR@5 and HR@5 on All the Five Datasets

Models	ColdRec-1		ColdRec-2		ColdRec-3		ColdRec-4		MovieLens	
	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5
UAF-Random	0.2562	0.4346	0.3917	0.5796	0.1172	0.2158	0.2123	0.3587	0.6905	0.9008
UAF-Gumbel	0.2594	0.4378	0.3940	0.5824	0.1190	**0.2212**	**0.2151**	**0.3640**	0.6942	0.9053
UAF-RL	0.2599	0.4386	**0.3942**	0.5837	0.1196	0.2197	0.2142	0.3619	**0.6959**	**0.9075**
UAF-Soft	**0.2600**	**0.4397**	0.3939	**0.5841**	**0.1209**	0.2189	0.2140	0.3613	0.6936	0.9041

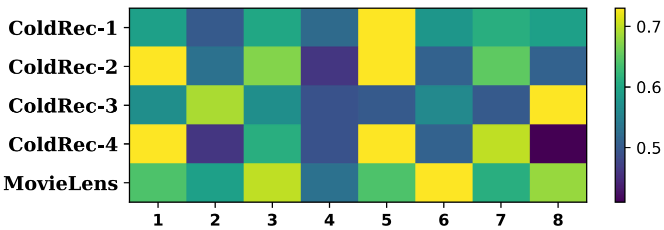


Fig. 3. Visualization of the policy. x-axis denotes the residual blocks (i.e., every two CNN layers) from 1st to 8th. The color means the rate of utilization of the fine-tuned residual blocks for all users in average. i.e., yellow means a higher utilization rate, whereas blue denotes a lower rate.

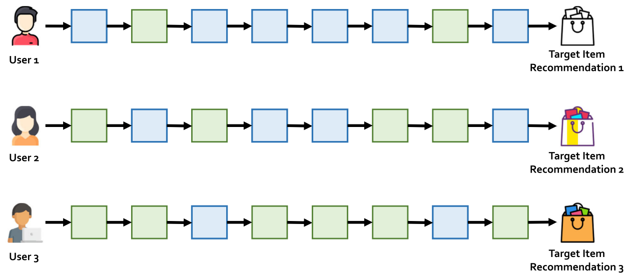


Fig. 4. Case study of three randomly chosen users in ColdRec-1. We regard *User 1* with 10 interacted items as a short-range user, *User 2* with 20 interacted items as a medium-range user, and *User 3* with 50 interacted items as a long-range user. The blue color represents the pre-trained layers, while the green color denotes the fine-tuned layers.

5.3.1 Random Policy

In order to verify the effectiveness of the policy network, we compare it with a random policy network, termed as UAF-Random, and the results are shown in the Table 4. The random policy network is implemented with a Bernoulli random distribution, which assigns 0.5 to the probability that the policy $\mathbf{I}_l(\mathbf{E}^u)$ equals 0 or 1. Note that our random seeds are kept fixed during training and inference. Clearly, the results in Table 4 show that a well-optimized policy outperforms a random policy. Interestingly, we find that UAF-Random performs comparably or slightly better than Finetune-All on the five datasets. The reason is because UAF-Random is able to combine parameters from both the pre-training and fine-tuning networks, albeit with a random way. However, if we use evaluation seeds different from training, UAF-Random shows much worse results than Finetune-All.

5.3.2 Visualization of Policies

To better understand the fine-tuning policies of UAF, we visualize the utilization rates of the fine-tuned residual blocks in Fig. 3. We only depict UAF-Gumbel since UAF-RL and UAF-Soft show similar behaviors. The illustration indicates that different datasets have completely different fine-tuning policies. Clearly, the utilization rates of all fine-tuned residual blocks are always smaller than 1. This indicates UAF does not only utilize fine-tuned blocks, instead, it also takes advantage of the pre-trained blocks since the summation of utilization rates of them equals to 1. Hence, we conclude that UAF allows the fine-tuning model to automatically identify the right policy in determining which layers of the pre-trained model should be fine-tuned and which layers should have their parameters frozen on a per-user basis, which would be infeasible through a manual approach.

In addition, we have also done some case studies on specific users to observe the fine-tuning policies, and the results are shown in Fig. 4. We select three randomly chosen users in ColdRec-1 to conduct the case study, for example, *User 1* with 10 interacted items as a short-range user, *User 2* with 20 interacted items as a medium-range user, and *User 3* with 50 interacted items as a long-range user. Results show that *User 1* chooses 2 out of 8 pre-trained residual blocks to be fine-tuned, while *User 2* and *User 3* choose 4 and 6 out of 8 pre-trained residual blocks to be fine-tuned, respectively, indicating that different users have personalized fine-tuning policies, and it seems that users with fewer observed

TABLE 5
Results by Replacing the Policy Networks (NextItNet versus GRU) in Terms of MRR@5 and HR@5 on All the Five Datasets

Models	ColdRec-1		ColdRec-2		ColdRec-3		ColdRec-4		MovieLens	
	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5	MRR@5	HR@5
UAF-Gumbel	0.2594	0.4378	0.3940	0.5824	0.1190	0.2212	0.2151	**0.3640**	0.6942	0.9053
UAF-Gumbel-GRU	0.2591	0.4382	**0.3943**	0.5836	0.1186	0.2204	**0.2159**	0.3634	0.6948	0.9056
UAF-RL	0.2599	0.4386	0.3942	0.5837	0.1196	0.2197	0.2142	0.3619	**0.6959**	**0.9075**
UAF-RL-GRU	0.2597	0.4381	0.3940	0.5833	0.1199	0.2201	0.2146	0.3625	0.6955	0.9069
UAF-Soft	**0.2600**	**0.4397**	0.3939	0.5841	0.1209	0.2189	0.2140	0.3613	0.6936	0.9041
UAF-Soft-GRU	0.2598	0.4389	0.3941	**0.5845**	**0.1217**	**0.2220**	0.2151	0.3627	0.6941	0.9047

TABLE 6
Evaluation Results With 10% Training Data in Terms of MRR@5 and HR@5 on ColdRec-1 and ColdRec-2

Model	ColdRec-1		ColdRec-2	
	MRR@5	HR@5	MRR@5	HR@5
Finetune-Zero	0.1624	0.2970	0.2766	0.4362
Finetune-All	0.1857	0.3366	0.2915	0.4533
PeterRec	0.1999	0.3562	0.2938	0.4557
UAF-Gumbel	0.2120	0.3714	**0.3025**	0.4651
UAF-RL	**0.2125**	**0.3721**	0.3023	**0.4655**
UAF-Soft	0.2108	0.3704	0.3021	0.4649

TABLE 7
Evaluation Results With 50% Training Data in Terms of MRR@5 and HR@5 on ColdRec-1 and ColdRec-2

Model	ColdRec-1		ColdRec-2	
	MRR@5	HR@5	MRR@5	HR@5
Finetune-Zero	0.1906	0.3393	0.3307	0.5072
Finetune-All	0.2060	0.3589	0.3406	0.5183
PeterRec	0.2184	0.3728	0.3415	0.5199
UAF-Gumbel	0.2296	0.3952	0.3498	0.5317
UAF-RL	0.2317	0.3986	**0.3522**	**0.5330**
UAF-Soft	**0.2321**	**0.4012**	0.3506	0.5321

interactions tend to retain more parameters of the pre-trained model and users with more observed interactions tend to choose more layers to be fine-tuned.

5.3.3 *Policy Network With GRU*

As mentioned before, the architecture of our policy network can be implemented with other deep neural networks without any restrictions. In order to verify the flexibility of it, we replace the original lightweight NextItNet with a Gated Recurrent Unit (GRU) for the policy network, termed as UAF-Gumbel-GRU, UAF-RL-GRU and UAF-Soft-GRU, respectively. We report the results in the Table 5. It clearly shows that designing the policy network by GRU yields comparable performance.

5.4 Robustness Studies With Limited Training Data (RQ4)

Transfer learning is supposed to reduce the cost of labeling data. In this subsection, we investigate the effectiveness of UAF with limited training samples. Specifically, we randomly choose 10% and 50% of the training examples in the target domain to fine-tune our models and the baselines. Tables 6 and 7 summarize the results of all models on ColdRec-1 and ColdRec-2. As shown, our methods (including UAF-Gumbel, UAF-RL and UAF-Soft) achieve notably better recommendation results than the Finetune-All and PeterRec. For example, the UAF-Gumbel method increases the MRR@5 metric by up to 6.1% on ColdRec-1 with 10% of the training examples in the target domain, compared with PeterRec. Therefore, we conclude that the proposed UAF methods could be more powerful than these baselines when dealing with limited training data.

It is well-known that deep models are prone to overfitting when training data is insufficient. We here investigate the effectiveness of the proposed UAF methods against overfitting. We depict the learning curves of different methods in Fig. 5. As shown, we observe that UAF-Gumbel, UAF-RL and UAF-Soft prevent overfitting much better than the Finetune-All. For example, on ColdRec-1, the HR@5 of the standard fine-tuning (Finetune-All) starts to decrease sharply after 5 epochs, whereas UAF-Gumbel, UAF-RL and UAF-Soft keep improving until 9 epochs, achieving notably higher HR@5. These results show that our methods can prevent overfitting better than many other models since UAF utilizes both pre-trained and fine-tuned parameters, where pre-trained parameters will not suffer from the overfitting issue.

5.5 Adaptability Experiments on User Profile Prediction Tasks (RQ5)

User profiles are important features for generating accurate recommendations. In [14], authors showed the transfer learning framework such as PeterRec and Finetune-All can also be used for predicting user profiles, which are of great help to lessen cold-start user problem in recommendation tasks. In this subsection, we investigate the versatility of UAF in the user profile prediction task.

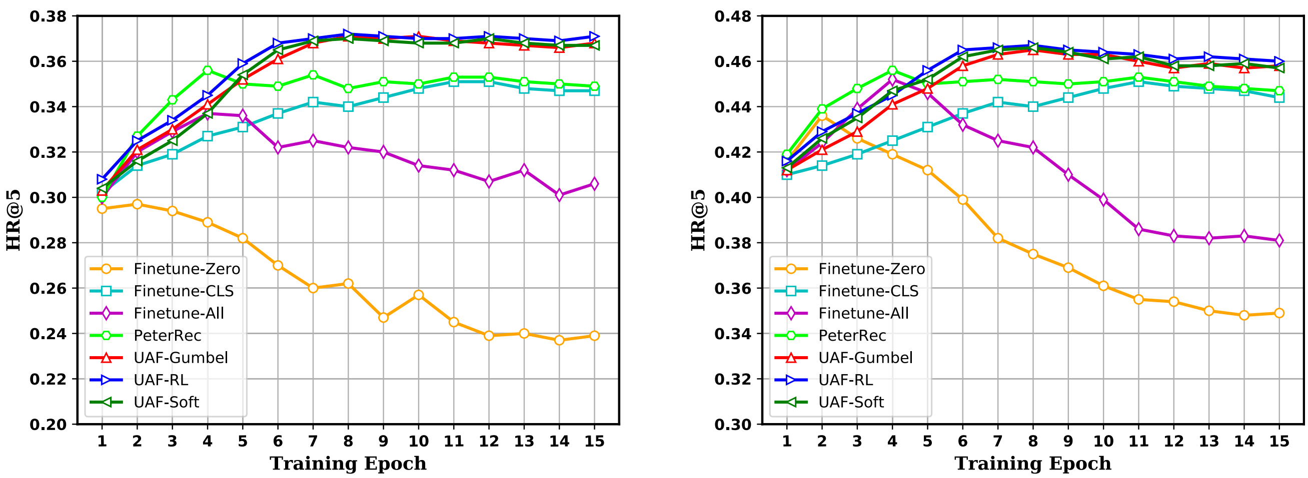


Fig. 5. HR@5 curves for different models on ColdRec-1 and ColdRec-2.

TABLE 8
Statistics of the Three User Profile Datasets

Domain	GenEst		AgeEst		LifeEst	
	Instances	Items&Classes	Instances	Items&Classes	Instances	Items&Classes
$\mathcal{S}$	1.55M	646K	1.55M	646K	1.08M	488K
$\mathcal{T}$	1.55M	2	1.55M	6	1.08M	8

We reported the total number of instances and items in source domain, and instances and classes in target domain in each dataset. Note that $K = 1000$ *and* $M = 1000K$.

TABLE 9
Results in Terms of Accuracy and F1 Regarding User Profile Prediction

Model	GenEst		AgeEst		LifeEst	
	Accuracy	F1	Accuracy	F1	Accuracy	F1
FinetuneCLS	0.8882	0.7770	0.5717	0.2747	0.5165	0.2471
FinetuneAll	0.9042	0.8205	0.6221	0.3457	0.5227	0.2670
PeterRec	0.9040	0.8184	0.6163	0.3338	0.5342	0.2616
UAF-Gumbel	**0.9051**	**0.8220**	0.6243	0.3481	**0.5491**	**0.2733**
UAF-RL	0.9049	0.8217	0.6250	0.3487	0.5478	0.2725
UAF-Soft	0.9048	0.8215	**0.6259**	**0.3496**	0.5484	0.2728

Following [14], we predict three types of user profile information (i.e., gender, age and life status). The statistics of the three user profile datasets are summarized in Table 8. Specifically, each instance in GenEst is a user and his gender (male or female) label obtained by the registration information. Similar to GenEst, each instance in AgeEst is a user and his age bracket label - one class represents 10 years. And each instance in LifeEst is a user and his life status label, e.g., single, married, pregnancy or parenting. The results regarding the user profile prediction tasks are shown in Table 9. We use classification accuracy (referred to as Accuracy) and F1 score (referred to as F1) as the evaluation metrics. Similar conclusions can be made as discussed in Section 5.1. UAF with adaptive fine-tuning policies yields better results than baselines in all three datasets, which demonstrates the versatility of UAF while adapting it to different downstream tasks.

6 Conclusion

In this paper, we proposed a User-specific Adaptive Fine-tuning method (UAF) for the cross-domain recommendations. UAF introduces a policy network to automatically decide which layers of the pre-trained model should be fine-tuned and which layers should have their parameters frozen on a per-user basis. It combines the advantages of high-capacity pre-trained network and newly optimized fine-tuned network, leading to enhanced performance on the target task. Extensive experiments on five real-world cross-domain recommendation datasets showed that UAF not only exhibited significantly better prediction accuracy, but also was more robust to overfitting, especially when the target domain has scarce training examples.

Acknowledgments

Lei Chen and Fajie Yuan contribute equally in this paper.

References

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

[2] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, “A simple convolutional generative network for next item recommendation,” in *Proc. 12th ACM Int. Conf. Web Search Data Mining*, 2019, pp. 582–590.

[3] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[4] W.-C. Kang and J. McAuley , “Self-attentive sequential recommendation,” in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 197–206.

[5] J. Xu, X. He, and H. Li, “Deep learning for matching in search and recommendation,” *Found. TrendsÂ® Inf. Retrieval*, vol. 14, no. 2/3, pp. 102–288, 2020.

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?,” in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[7] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” 2018.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” 2018, *arXiv: 1810.04805*.

[9] Y. Guo, Y. Li, L. Wang, and T. Rosing, “AdaFilter: Adaptive filter fine-tuning for deep transfer learning,” 2019, *arXiv: 1911.09659*.

[10] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, “Efficient parametrization of multi-domain deep neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8119–8127.

[11] H. Kanagawa, H. Kobayashi, N. Shimizu, Y. Tagami, and T. Suzuki, “Cross-domain recommendation via deep domain adaptation,” in *Proc. Eur. Conf. Inf. Retrieval*, 2019, pp. 20–29.

[12] F. Yuan, L. Yao, and B. Benatallah, “DARec: Deep domain adaptation for cross-domain recommendation via transferring rating patterns,” 2019, *arXiv: 1905.10760*.

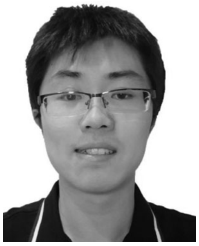
[13] G. Hu, Y. Zhang, and Q. Yang, “CoNet: Collaborative cross networks for cross-domain recommendation,” in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 667–676.

[14] F. Yuan, X. He, A. Karatzoglou, and L. Zhang, “Parameter-efficient transfer from sequential behaviors for user modeling and recommendation,” in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1469–1478.

[15] F. Yuan, G. Zhang, A. Karatzoglou, J. Jose, B. Kong, and Y. Li, “One person, one model, one world: Learning continual user representation without forgetting,” *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 696–705.

[16] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[17] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019, Art. no. 9.
[18] T. B. Brown *et al.*, "Language models are few-shot learners," 2020, *arXiv: 2005.14165*.
[19] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. V. D. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," 2016, *arXiv:1610.10099*.
[20] A. V. D. Oord *et al.*, "WaveNet: A generative model for raw audio," 2016, *arXiv:1609.03499*.
[21] J. Wang, Q. Liu, Z. Liu, and S. Wu, "Towards accurate and interpretable sequential prediction: A CNN & attention-based feature extractor," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1703–1712.
[22] F. Yuan *et al.* "Future data helps training: Modeling future contexts for session-based recommendation," in *Proc. Web Conf.*, 2020, pp. 303–313.
[23] Y. Sun, F. Yuan, M. Yang, G. Wei, Z. Zhao, and D. Liu, "A generic network compression framework for sequential recommender systems," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1299–1308.
[24] J. Wang *et al.*, "StackRec: Efficient training of very deep sequential recommender models by iterative stacking," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 357–366.
[25] L. Chen, F. Yuan, J. Yang, X. Ao, C. Li, and M. Yang, "A user-adaptive layer selection framework for very deep sequential recommender models," in *Proc. AAAI Conf. Artif. Intell.*, 2021.
[26] F. Sun *et al.* "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 1441–1450.
[27] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 17–22.
[28] J. Tang and K. Wang, "Personalized Top-N sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining*, 2018, pp. 565–573.
[29] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 843–852.
[30] G. D. S. P. Moreira, D. Jannach, and A. M. D. Cunha, "Contextual hybrid session-based news recommendation with recurrent neural networks," *IEEE Access*, vol. 7, pp. 169 185–169 203, 2019.
[31] Z. Zeng *et al.*, "Knowledge transfer via pre-training for recommendation: A review and prospect," *Front. Big Data*, vol. 4, 2021.
[32] M. M. Khan, R. Ibrahim, and I. Ghani, "Cross domain recommender systems: A systematic literature review," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–34, 2017.
[33] B. Tran, M. Karimzadehgan, R. K. Pasumarthi, M. Bendersky, and D. Metzler, "Domain adaptation for enterprise email search," in *Proc. 42nd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2019, pp. 25–34.
[34] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 343–351.
[35] P. Li and A. Tuzhilin, "DDTCDR: Deep dual transfer cross domain recommendation," in *Proc. 13th Int. Conf. Web Search Data Mining*, 2020, pp. 331–339.
[36] Y. Zhang *et al.*, "Learning personalized itemset mapping for cross-domain recommendation," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 2561–2567.
[37] C. Zhao, C. Li, R. Xiao, H. Deng, and A. Sun, "CATN: Cross-domain recommendation for cold-start users via aspect transfer network," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 229–238.
[38] Z. Ren, L. Zhao, J. Ma, and M. de Rijke, "Mixed information flow for cross-domain sequential recommendations," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, 2020.
[39] Y. Ni *et al.*, "Perceive your users in depth: Learning universal user representations from multiple e-commerce tasks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 596–605.
[40] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5754–5764.
[41] L. Bertinetto, J. F. Henriques, J. Valmadre, P. Torr, and A. Vedaldi, "Learning feed-forward one-shot learners," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 523–531.
[42] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, "SpotTune: Transfer learning through adaptive fine-tuning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4800–4809.
[43] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme , "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, 2009, pp. 452–461.
[44] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," 2016, *arXiv:1611.00712*.
[45] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-softmax," 2016, *arXiv:1611.01144*.
[46] Z. Wu *et al.*, "BlockDrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8817–8826.
[47] Y. Bengio, "Deep learning of representations: Looking forward," in *Proc. Int. Conf. Statist. Lang. Speech Process.*, 2013, pp. 1–37.
[48] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1179–1195.
[49] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, pp. 229–256, 1992.
[50] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3994–4003.
[51] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," 2015, *arXiv:1502.02791*.
[52] R. Caruana, "Multitask learning," *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.
[53] X. He, Z. He, X. Du, and T. Chua, "Adversarial personalized ranking for recommendation," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 355–364.

Lei Chen received the BS degree in computer science and technology from the China University of Mining and Technology, Xuzhou, China, in 2019. He is currently working toward the master degree in the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His research interests include natural language processing, recommendation systems, and information retrieval.



Fajie Yuan received the PhD degree from the University of Glasgow, Glasgow, Scotland. He is currently an assistant professor with Westlake University. Prior to that, he was a senior AI researcher with Tencent working on recommender systems. His main research interests include deep learning and transfer learning and their applications in recommender systems.

Jiaxi Yang received the BS degree from the Huazhong University of Science and Technology, Wuhan, China, in 2021. He is currently working toward the PhD degree in the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. His research interests include recommendation systems and information retrieval.

Xiangnan He received the PhD degree in computer science from the National University of Singapore (NUS), Singapore. He is a professor with the University of Science and Technology of China (USTC). His research interests span information retrieval, data mining, and multi-media analytics. He has more than 80 publications that appeared in several top conferences such as SIGIR, WWW, and MM, and journals including the *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, and *IEEE Transactions on Multimedia*. His work has received the Best Paper Award Honorable Mention in WWW 2018 and ACM SIGIR 2016. He is in the editorial board of journals including *Frontiers in Big Data*, *AI Open* etc. Moreover, he has served as the PC chair of CCIS 2019 and SPC/PC member for several top conferences including SIGIR, WWW, KDD, MM, WSDM, ICML etc., and the regular reviewer for journals including the *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, IEEE Transactions on Multimedia, etc.

Chengming Li received the BS and MS degrees in computer application technology from the Dalian University of Technology, Dalian, China, in 2009 and 2011, respectively, and the PhD degree from the Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan, in 2015. He currently is an associate professor with the School of Intelligent Systems Engineering, Sun Yat-Sen University. His research interests include data mining, network security, and big data.

Min Yang received the BS degree from Sichuan University, Chengdu, China, in 2012, and the PhD degree from the University of Hong Kong, Hong Kong, in February 2017. She is currently an associate professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Science. Her current research interests include recommendation systems, deep learning, and natural language processing.