PRELIMINARY VERSION: DO NOT CITE
The AAAI Digital Library will contain the published
version some time after the conference

A User-Adaptive Layer Selection Framework for Very Deep Sequential Recommender Models

Lei Chen[1,2*], Fajie Yuan[3*], Jiaxi Yang[4], Xiang Ao[5], Chengming Li[1†], Min Yang[1†].

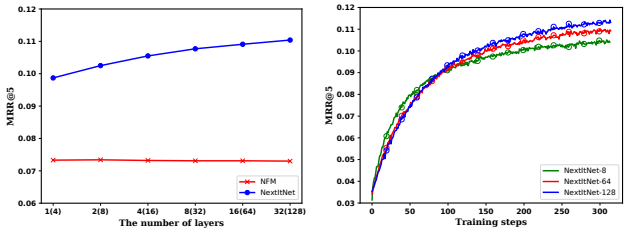
[1]Shenzhen Key Laboratory for High Performance Data Mining,
Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences [3]Tencent
[4]Huazhong University of Science and Technology
[5]Institute of Computing Technology, Chinese Academy of Sciences
{lei.chen, cm.li, min.yang}@siat.ac.cn, fajieyuan@tencent.com, yangjiaxi@hust.edu.cn, aoxiang@ict.ac.cn

Abstract

Sequential recommender systems (SRS) have become a research hotspot in recent studies. Because of the requirement in capturing user's dynamic interests, sequential neural network based recommender models often need to be stacked with more hidden layers (e.g., up to 100 layers) compared with standard collaborative filtering methods. However, the high network latency has become the main obstacle when deploying very deep recommender models into a production environment. In this paper, we argue that the typical prediction framework that treats all users equally during the inference phase is inefficient in running time, as well as sub-optimal in accuracy. To resolve such an issue, we present SkipRec, an adaptive inference framework by learning to skip inactive hidden layers on a per-user basis. Specifically, we devise a policy network to automatically determine which layers should be retained and which layers are allowed to be skipped, so as to achieve user-specific decisions. To derive the optimal skipping policy, we propose using gumbel softmax and reinforcement learning to solve the non-differentiable problem during backpropagation. We perform extensive experiments on three real-world recommendation datasets, and demonstrate that SkipRec attains comparable or better accuracy with much less inference time.

1 Introduction

The past decade has seen a remarkable progress in deep learning and their applications in recommender systems (RS). A variety of neural network models with larger and deeper architectures are proposed to model user-item behavior interactions from online systems. Among them, sequential recommendation models (Hidasi et al. 2016; Yuan et al. 2019; Kang and McAuley 2018) have become especially popular due to their powerful capacity in modeling user's dynamic interests. In addition, sequential models are well-suited to do self-supervised learning by predicting the next interaction in the sequence (Yuan et al. 2020b,c), and thereby require less feature engineering and manually labeled data.



(a) (b)

Figure 1: Recommendation accuracy (MRR@5) w.r.t. the number of layers of NFM and NextItNet on the Weishi (Sun et al. 2020) dataset. On (a), the left(right) label of x-axis represents the number of layers for NFM and NextItNet, respectively. (a) Compared with NFM, deepening NextItNet results in significantly better accuracy; (b) the 128-layer NextItNet converges better than other shallower NextItNets. Experimental settings are given in Section 4.

In fact, compared with standard collaborative filtering (CF) methods, sequential recommender models enjoy excellent model expressivity by stacking very deep layers. As shown in Figure 1, we compare two representative recommendation models, namely, neural factorization machines (NFM) (He and Chua 2017) for standard CF and NextItNet (Yuan et al. 2019) (with a slight change as described in Section 3.3) for sequential recommendations. As shown, on (a), we clearly observe that NFM yields no accuracy gain by deepening the number of network layers, and it performs largely worse than NextItNet. In contrast, the performance of NextItNet is gradually improved by stacking more layers. To our surprise, NextItNet converges the best with up to 128 layer, achieving around 1% improvement on MRR@5 over the same model with 64 layers, as shown in (b).

However, as the network becomes deeper, a real problem arises, that is, the inference cost largely increases, leading to an inevitable time delay for online service. In this paper, we argue that users in recommender systems are unique (i.e., personalized), passing all of them through hidden layers of the same depth is non-optimal and computationally inefficient. Instead, as shown in Figure 2, we believe that for many 'easy' users (or user input sequences), neural network models with a few hidden layers are already expressive enough, while for these 'hard' users, more layers are often necessary

*The two authors contributed equally to this paper.
†Chengming Li and Min Yang are co-corresponding authors.

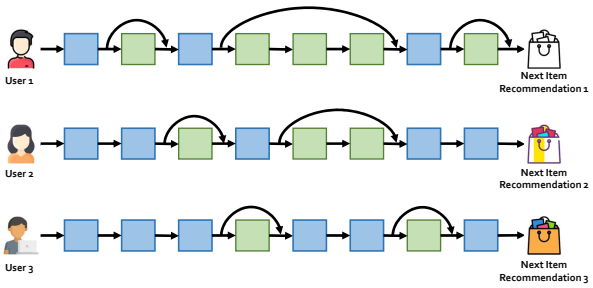


Figure 2: Skipping inactive layers on a per-user basis. For user 1, five of eight layers can be skipped, which is regarded as an easy user, As a comparison, user 3 is thought of as a relatively hard user since only two layers are allowed to skip.

so as to capture complex preferences. Unfortunately, to the best of our knowledge, thus far all existing deep neural network based recommender models assign fixed structures for users when generating recommendations.

To address the aforementioned issue, we propose SkipRec, an adaptive inference framework for deep sequential recommender models, which defines the network structure adaptively on a per-user basis. Specifically, we devise a policy network to automatically determine which layers in the backbone network should be retained and which layers are skipped, so as to obtain the user-specific decision in SRS. SkipRec is a general network depth selection framework which directly applies to a broad range of deep recommendation models, such as NextItNet (Yuan et al. 2019) and SASRec (Kang and McAuley 2018).

We summarize our main contributions as follows:

- We are the first to emphasize the unnecessity in executing the same number of hidden layers for all users in a deep recommender model.
- We propose SkipRec, a user-specific depth selection framework where the number of network layers can be selected on a per-user basis. SkipRec enables each user to have their own skipping policies, which is the first personalized depth selection method for the recommendation task.
- We propose SkipRec-Gumbel and SkipRec-RL to derive the optimal skipping policy (skipping or retaining) without suffering from the non-differentiable problem during backpropagation.
- Extensive experiments show that the proposed SkipRec attains competitive or better performance with less inference time in three real-world SRS datasets.

2 Related Work

Sequential recommender systems (SRS), a.k.a. session-based RS, take user's past behaviors as input and generate recommendations by analyzing, modeling and representing these behaviors. In this paper, we simply categorize SRS into conventional SRS and deep learning (DL) based SRS.

Early work of SRS mostly rely on the Markov Chain (MC) (Shani, Heckerman, and Brafman 2005) and factorization models (Rendle, Freudenthaler, and Schmidt-Thieme 2010). Specifically, Shani, Heckerman, and Brafman (2005) introduced Markov Decision Processes (MDPs) based methods to model the sequential nature of the recommendation process. Rendle, Freudenthaler, and Schmidt-Thieme (2010) proposed factorized personalized Markov chains to combine the transition information between the adjacent behaviors and the item latent factors for the next item recommendation. Another representative work in Wang et al. introduced a hierarchical representation method to capture both sequential patterns and the general taste of users by involving representations and sequential transactions in prediction. The major problems of these methods are that they merely characterize the local sequential patterns of user behavioral sequence and adopt static representations for user preference. Thereby, such methods often show poor performance when modeling complicated and dynamic relations in the sequential session data.

Driven by the great success of DL, recurrent neural networks (RNNs) (Hidasi et al. 2016; Wu et al. 2017; Ying et al. 2018) have been sucessfully adapted in SRS and shown obvious improvements compared with convential non-DL models. For example, Hidasi et al. (2016) proposed GRU4Rec, which used the gated recurrent unit (GRU) to model the short-term preference based on user's previous purchasing or clicking behaviors. Subsequently, a variety of RNN variants have been proposed to improve the performance of the SRS tasks, such as personalized SRS (Ying et al. 2018), content- and context-based SRS (Gu et al. 2016; Smirnova and Vasile 2017). Despite the effectiveness of RNN-based methods, they rely heavily on hidden states of entire past items, neglecting the parallel processing resources (e.g., GPU and TPU). Convolutional Neural Networks (CNNs) and self-attention models are proposed to mitigate such limitations (Tang and Wang 2018; Yuan et al. 2019, 2020a; Sun et al. 2019). Tang and Wang (2018) developed a convolutional sequence embedding recommendation method (Caser) for SRS, which embeds a sequence of user-item interaction into an "image" and learn sequential patterns as local features of the image using wide convolutional filters. Yuan et al. (2019) developed a deep CNN-based sequential recommendation model (NextItNet) to learn the complex conditional distribution in both short and long-range item sequences. In parallel, self-attention based models, such as SASRec (Kang and McAuley 2018) and BERT4Rec (Sun et al. 2019) also obtained competitive results.

As revealed by previous works (Yuan et al. 2019; Sun et al. 2020; Yuan et al. 2020a; Bachlechner et al. 2020), CNNs and self-attention models can perform better than RNN-based sequential models since their performance can be boosted by stacking more hidden layers with the residual block architecture. However, very deep networks are always accompanied with increased prediction cost and latency (Wang et al. 2018). Distinct from existing work, we propose a user-specific adaptive network to dynamically select which layers of a CNN or self-attention model should be skipped during inference based on user's historical behaviors.

3 Our Methodology

3.1 Problem Definition

Given a sequence of historical user behaviors $X^u = [x_1^u, x_2^u, \dots x_t^u]$, where x_t^u denotes the t-th interacted item of user u, the goal of sequential recommendation (SR) is to predict item x_{t+1}^u that the user will interact with at time $t+1$. Note that it is straightforward to change the above task setting into a basket-based setting by replacing each item x_i^u with an item subset (i.e., basket) (Yu et al. 2016). For simplicity, we keep the next-item recommendation as our major task setting. Meanwhile, given that users often merely care about the first few items, the top-N items are recommended.

3.2 The Overall Architecture

In the paper, we describe SkipRec using NextItNet-like (Yuan et al. 2019) learning algorithm as the backbone model given its superior recommendation performance. It is noteworthy that SkipRec is model-agnostic and potentially applicable for any SR model with a deep network architecure. Figure 3 illustrates the overall architecture of the proposed method. To be specific, SkipRec consists of two primary modules: backbone network and policy network. The backbone network is a slightly modified NextItNet model (described later), which is more powerful than the original version. We devise a policy network to automatically determine which layers in the backbone network should be retained and which layers should be skipped on a per-user basis, so as to truly achieve user-specific decision. Next, we will first recapitulate the backnone network and then introduce our policy network in detail.

3.3 Backbone Network

NextItNet is composed of a stack of holed convolutional layers, which are wrapped by a residual block structure every two layers. Specifically, each input item x^u is converted into an embedding vector $\mathbf{e}^u$, and the interaction sequence X^u is thereby represented by an embedding matrix $\mathbf{E}^u = [\mathbf{e}_1^u \dots \mathbf{e}_t^u]$. The item embeddings are then passed into a stack of dilated convolutional (DC) layers to learn feature vector $\mathbf{E}_l^u$ which is expected to capture the long-range dependencies. Here, l represents the l-th residual block in the backbone network. A residual block is employed the connect every two consecutive DC layers. Formally, the l-th residual block with the DC operation is formalized as:

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

where $\mathbf{E}_{l\ 1}^u$ and $\mathbf{E}_l^u$ are input and output of the l-th residual block considered. $\lambda \times \mathcal{F}_l(\mathbf{E}_{l\ 1}^u) + \mathbf{E}_{l\ 1}^u$ is a shortcut connection by element-wise addition. Similar to (Wang et al. 2020; Bachlechner et al. 2020), we add a learnable coefficient λ (initialized with zero[1]) to the residual mappings $\mathcal{F}_l(\mathbf{E}_{l\ 1}^u)$, so that the model can stack more layers, even more than 100 layers, and get better results than the standard version with λ as 1. $\mathcal{F}_l(\mathbf{E}_{l\ 1}^u)$ represents the residual mapping, which is defined as:

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

[1]We empirically find that initializing λ to zero always helps the model converge better and faster than other bigger values.

where ψ_1 and ψ_2 represent the casual convolution operations. $\mathbf{LN_1}$ and $\mathbf{LN_2}$ represent layer normalization functions. σ is the ReLU activation function.

Finally, a softmax output layer is applied to predict the probability distribution for the next item x_{t+1}^u:

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

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

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

$$p(X^u; \Omega) = \prod_{i=2}^{t} p\left(x_i^u | x_{1:i\ \ 1}^u, \theta\right) p(x_1^u) \tag{4}$$

where $p\left(x_i^u | x_{1:i\ \ 1}^u; \Omega\right)$ is the predicted probability for the i-th 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.

3.4 Policy Network

In this subsection, we design a user-specific policy network to output a binary policy vector $\mathbf{I}_l(\mathbf{E}^u)$, representing the actions to retain or skip the l-th residual block in the backbone network based on user sequence $\mathbf{E}^u$. In particular, the policy network is implemented 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 any deep neural networks, e.g., a RNN model.

Specifically, for the l-th residual block in the backbone network, we learn a policy $\mathbf{I}_l(\mathbf{E}^u)$ through the policy network to decide whether or not to skip $\mathbf{E}_l^u$ during training. The adaptive output $\mathbf{E}_{l\ adaptive}^u$ of the l-th residual block in SkipRec is computed as:

$$\mathbf{E}_{l\ adaptive}^u = \mathbf{I}_l(\mathbf{E}^u)\mathbf{E}_l^u + (1 - \mathbf{I}_l(\mathbf{E}^u))\,\mathbf{E}_{l\ 1\ adaptive}^u \tag{5}$$

where $\mathbf{I}_l(\mathbf{E}^u)$ is a binary variable indicating whether the residual block $\mathbf{E}_l^u$ could be retained or skipped based on user sequence $\mathbf{E}^u$. During training, the input user sequence can either retain the current block $\mathbf{E}_l^u$ or skip it by directly using the output of the previous residual block $\mathbf{E}_{l\ 1\ adaptive}^u$ in SkipRec. $\mathbf{I}_l(\mathbf{E}^u)$ draws samples from a discrete distribution of two classes (retain or skip), which could be parameterized by the output of a small-sized policy network. To be more specific, the l-th residual block in backbone network is skipped if $\mathbf{I}_l(\mathbf{E}^u) = 0$; otherwise, the l-th residual block is retained.

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 two methods to generate the actions (retain or skip) from a discrete distribution: the Gumbeling Softmax sampling method (Maddison, Mnih, and Teh 2017) and the reinforcement learning method (Rennie et al. 2017).

Gumbeling Softmax Sampling We employ the Gumbeling Softmax sampling method (Maddison, Mnih, and Teh 2017) to produce the actions (retain or skip) from a discrete distribution (called **SkipRec-Gumbel**). Specifically,

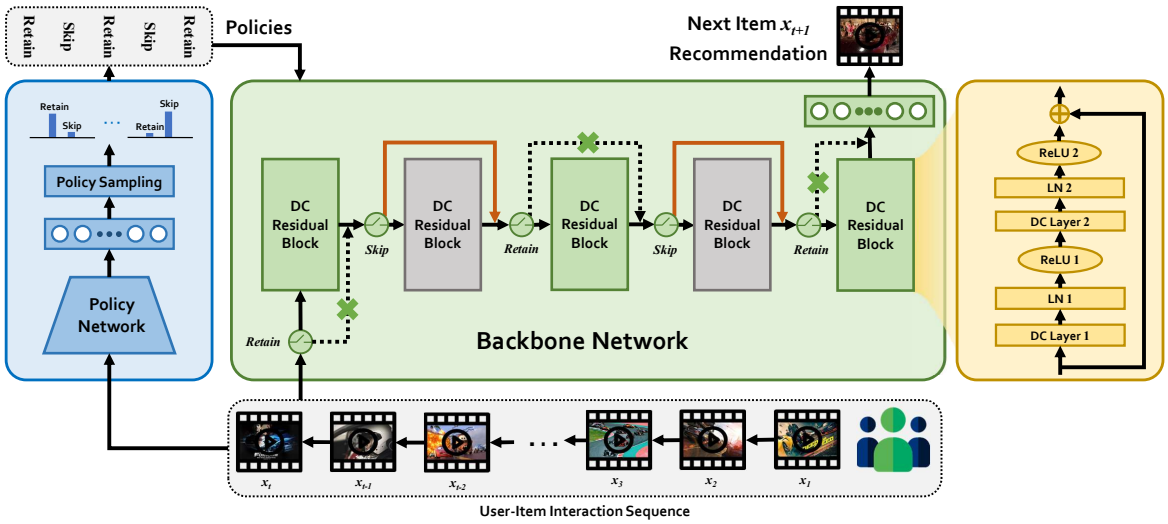


Figure 3: Model architecture. The proposed SkipRec consists of two primary modules: backbone network and policy network. In backbone network, we adopt a modified NextItNet model to capture important information in user-item interaction sequences. In policy network, we learn a user-specific adaptive skipping strategy to automatically determine which layers in the backbone network should be retained and which layers should be skipped on a per-user basis.

SkipRec-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 retaining or skipping actions. That is, each residual block is associated with two scalars π_1 & π_2 that correspond to final output of policy network.

$$\mathbf{z} = \text{one hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right) \tag{6}$$

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. In particular, we sample the $Gumbel(0, 1)$ distribution using inverse transform sampling by drawing u from a uniform distribution (Jang, Gu, and Poole 2016), i.e. $u \sim Uniform(0, 1)$ and compute $g = -\log(-\log(u))$.

The $\arg\max$ operation in Eq. (6) is non-differentiable, but we can resort to the Gumbel Softmax distribution, which adopts softmax as a continuous relaxation to $\arg\max$ in order to alleviate the non-differentiable problem. We relax the one-hot encoding of $\mathbf{z}$ to a real-valued vector $\boldsymbol{\alpha}$ using:

$$\alpha_i = \frac{\exp\left(\left(\log \pi_i + g_i\right)/\tau\right)}{\sum_{j=1}^{k} \exp\left(\left(\log \pi_j + g_j\right)/\tau\right)} \quad \text{for } i = 1, \ldots, k \tag{7}$$

where τ is a temperature parameter to control the discreteness of the output vector $\boldsymbol{\alpha}$. Here we set τ to 10 by default.

We can solve the non-differentiable problem by sampling the skipping policy $\mathbf{I}_l(\mathbf{E}^u)$ from the Gumbel Softmax distribution, since the Gumbel Softmax distribution is smooth when $\tau > 0$ and thus has well-defined gradients in terms of the parameters π_i. Similar to (Wu et al. 2018; Guo et al. 2019), we generate all skip/retain policies for all residual blocks at once for the trade-off of efficiency and accuracy. During the forward pass, we sample the skipping policy $\mathbf{I}_l(\mathbf{E}^u)$ using Eq. (6) for the l-th block. As for the backward pass, we are able to estimating the gradients of the discrete samples by computing the gradients of the continuous softmax relaxation in Eq. (7).

By using the above approach, we are able to optimize the SkipRec in a differentiable way and obtain the policy regarding which layers in backbone network should be skipped. The policy network is simultaneously trained with the backbone network in an end-to-end way. We use the standard cross-entropy objective function as in the original NextItNet model.

Reinforcement Learning As an alternative to the SkipRec-Gumbel method, we learn the policy network by employing the reinforcement learning algorithm (called **SkipRec-RL**). The main idea is to learn the policy network that outputs the posterior probabilities of all the binary decisions for keeping or skipping each block in the backbone network. The policy network is optimized via curriculum learning (Bengio 2013; Wu et al. 2018) to maximize a reward that incentivizes the use of blocks as few as possible while maintaining the prediction performance. In this regard, we can consider the potential trade-offs between computational cost and prediction accuracy.

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

$$\mathbf{R}(\mathcal{A}) = \begin{cases} 1 - \left(\sum_l \mathbb{1}(\mathcal{A}_l)/N\right)^2 & \text{if correct} \\ -\gamma & \text{otherwise} \end{cases} \tag{8}$$

where $\mathbb{1}(\cdot)$ is an indicator function, $\mathcal{A}_l$ represents the skip or retain policy applied on the l-th block, γ is a hyper parameter to penalize the wrong policy. The motivation behind the Eq. 8 is two fold: (1) enabling the policy net-

work to control the backbone network so as to generate well-recommended items; (2) achieving a significant inference speedup to meet the requirement of online service. Specifically, $\left(\sum_l \mathbb{1}(\mathcal{A}_l)/N\right)^2$ measures the percentage of blocks that are utilized. When a correct recommendation is produced, we incentivize the skipped block by assigning a larger reward to a policy which utilizes fewer blocks. In addition, we penalize incorrect recommendations with γ, which determines 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) (Rennie et al. 2017), which is a form of REINFORCE (Williams 1992) algorithm, for model training. Instead of estimating a "baseline" to normalize the rewards and shrink variance, SCST applies the output of its own test-time inference algorithm so as to normalize the rewards. In details, the exploration action $\mathcal{A}_l^s$ is obtained by sampling from the categorical distribution $p(\pi_i)$ through modeling the user-item interaction 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 optimize the parameters of policy network, we minimize the below SCST loss:

$$\mathcal{L}_{RL} = -\sum_{l=1}^{N} \log p(\mathcal{A}_l^s) \left(\mathbf{R}(\mathcal{A}_l^s) - \mathbf{R}(\hat{\mathcal{A}}_l)\right) \tag{9}$$

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 testing time estimation under the current model, the SCST is encouraged to boost the performance of the model under the inference algorithm at testing time (Rennie et al. 2017). At inference stage, we obtain the actual skipping 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 backbone network in an end-to-end way and optimize the weighted-sum of the cross-entropy loss and SCST loss in the SkipRec-RL method:

$$\mathcal{L} = \mathcal{L}_{CE} + \beta \mathcal{L}_{RL} \tag{10}$$

where $\mathcal{L}_{CE}$ and $\mathcal{L}_{RL}$ represent the standard cross-entropy loss and the SCST loss respectively, and β is a hyper parameter to control the weight of the SCST loss, which is set to 1 in our experiments.

Dataset	#items	#interactions	#sequences	Length t
Weishi	66K	10M	1,048,575	10
ML20	54K	27.7M	1,491,478	20
ML100	54K	27.7M	457,350	100

Table 1: Dataset statistics (after pre-processing). "K" and "M" are short for thousand and million, respectively.

3.5 Training Procedure

In the training procedure, we propose two ways to train our SkipRec model: one-stage and two-stage. In one-stage training, the parameters of the policy network and the backbone network are initialized randomly and trained jointly. While in the two-stage training, we first pre-train the NextItNet model with training data and initialize the backbone network with pre-trained parameters, which can help the backbone network obtain better feature representation ability at the beginning and make the training of the policy network more conducive.

4 Experimental Setup

4.1 Experimental Datasets

We conduct extensive experiments on three real-world datasets: ML20, ML100, and Weishi (Sun et al. 2020). The statistics of these three datasets are reported in Table 1.

- ML20 and ML100: The MovieLens[2] dataset is widely used for both standard collaborative filtering and sequential recommendations, which consists of approximately 280,000 users, 58,000 videos (full movies and clips) and 27 million time-stamped user-item interactions. Similar to (Rendle et al. 2009; Sun et al. 2020), we remove the users with less than 10 items and the interactions with less than 5 users so as to alleviate the impact of cold users and items, respectively. Afterwards, the maximum length of each interaction sequence is set to be t. We split the sequences longer than t into multiple sub-sequences, while the sequences shorter than t are padded with zero in the beginning of each sequence, similar to (Yuan et al. 2019). In this paper, we set t to be 20 (denoted as ML20) and 100 (denoted as ML100) as short- and long-range sequences, respectively.
- Weishi[3]: This is a short-video (around 20 seconds per video) recommendation dataset provided by Tencent (Sun et al. 2020). It contains more than 60,000 videos and around 1 million users. Following (Sun et al. 2020), we handle it as a short-range sequential dataset by extracting the latest 10 videos per user.

4.2 Baselines and Evaluation

While the main purpose of this work is to demonstrate the effectiveness of SkipRec, we compare its accuracy with several other well-known recommendation methods for reference, including NFM (He and Chua 2017), GRU4Rec (Tan, Xu, and Liu 2016), Caser (Tang and Wang 2018), NextItNet (Yuan et al. 2019) and its advanced variant as described in Section 3.3, termed as NextItNet+ in the remaining. Note that NFM is also suited for the sequential recommendation settings by treating user interactions as common features. As for GRU4Rec, we report results with the autoregressive training method (Yuan et al. 2020a) for fair comparison with NextItNet.

We measure the performance of top-N accuracy following previous works including HR@N (Hit Ratio) and MRR@N (Mean Reciprocal Rank) (Hidasi et al. 2016; Yuan et al. 2019). Here N is set to 5 and 20. We report the inference speedup compared to the backnone network NextItNet+ to reflect its inference efficiency. Note that we only evaluate the prediction accuracy of the last item in each user-item interaction sequence in testing set, similarly to (Yuan et al. 2019; Sun et al. 2020).

[2]https://grouplens.org/datasets/movielens/

[3]https://weishi.qq.com

4.3 Implementation Details

We divide the number of user sessions in each dataset into training (80%), validation (5%) and test (15%) sets. For fair comparison with previous works, SkipRec adopts the same values as NextItNet+ for the shared hyper-parameters. For policy network, we use dilation factors of $\{1, 2, 4, 8\}$ (4 layers or 2 residual blocks), which is a lightweight neural network. We use a batch size (denoted by b) of 256 and dilation factors (denoted by l) of $16 \times \{1, 2, 4, 8\}$ (64 layers or 32 residual blocks) for the backbone network on Weishi and ML20. Note that we use 64 layers on Weishi and ML20 so as to verify that SkipRec works well even the model has not fulfill its total capacity since the best performance is achieved up to 128 layers. On ML100, we set b to 256 and l to $24 \times \{1, 2, 4, 8\}$ (96 layers). The size of item embeddings d is set to be 256 for all datasets. For other baselines, we use the same item embedding size and hidden dimensions. We employ Adam optimizer to train all models with learning rate η of 0.001 given their good behaviors on the validation set. All the experiments are implemented in TensorFlow and trained on a single TITAN RTX GPU.

5 Experimental Results

5.1 Quantitative Evaluation

Table 2 reports the results of SkipRec and all baselines on the three datasets. From the results, we can make the following observations. First, we observe that NextItNet performs better than NFM, GRU4Rec and Caser with substantial improvements in terms of recommendation accuracy among the three SRS datasets, which is consistent with the previous work (Yuan et al. 2019; Sun et al. 2020). Second, the modified NextItNet+ can achieve better recommendation accuracy than NextItNet across all datasets with exactly the same hyper-parameters. Third, SkipRec with the proposed Gumbel Softmax sampling method (SkipRec-Gumbel) and reinforcement learning method (SkipRec-RL) attains comparable or better recommendation accuracy, especially when we pre-train the backbone network (i.e., Gumbel w/pre and RL w/pre) in advance. For example, on Weishi and ML20, SkipRec-Gumbel with pre-training obtains 2.7% and 3.5% in terms of MRR@5, respectively. More importantly, SkipRec achieves a notable inference speedup relative to NextItNet+ (e.g., 2.01× and 2.30× inference speedup with SkipRec-RL with pre-training on Weishi and ML20).

5.2 Ablation Studies on the Policy Network

In the remaining ablation studies, we only report partial experiments for saving space. For example, we just report SkipRec with the pre-trained backbone network in this subsection due to similar conclusions.

Random Policy In order to verify the effectiveness of the policy network, we compare it with a random policy network, and the results are shown in the Table 3. The random policy network is implemented with a Bernoulli random distribution which can make the probability that the policy $\mathbf{I}_l(\mathbf{E}^u)$ equals 0 or 1 as 0.5. The results demonstrate that a well-optimized policy largely performs better than a random policy. Moreover, a random policy degrades the original backbone network clearly since the network connectivity has been broken.

Policy Network with GRU As mentioned before, the architecture of our policy network can be implemented with any deep neural networks without any restrictions. In order to verify the flexibility of the policy network, we replace the original lightweight NextItNet with a Gated Recurrent Unit (GRU) for the policy network, and the results are shown in the Table 4. The results show that designing the policy network by a Gated Recurrent Unit (GRU) can also achieve comparable performance.

Visualization of Policies To better understand the skipping policies learned by policy network, we visualize them on the three datasets in Figure 4. For saving space, we only show SkipRec-Gumbel while simply omitting SkipRec-RL due to their similarities. The illustration shows that different datasets have different skipping policies. SkipRec allows NextItNet+ to automatically identify the right policy in determining which layers in the backbone network should be retained and which layers should be skipped on a per-user basis, which would be infeasible through a manual approach.

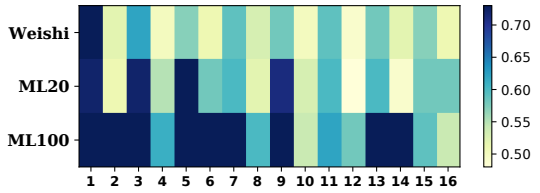


Figure 4: Visualization of the policy. x-axis denotes the residual blocks (i.e., every two CNN layers) from 1st to 16th. The color means the rate of utilization of residual blocks for all users in average. i.e., dark blue means a higher utilization rate, whereas light yellow denotes a lower rate.

5.3 Convergence Behavior Analysis

Here we investigate the convergence behaviors of SkipRec. In particular, we visualize the learning curves of NextItNet+ and proposed SkipRec-Gumbel & SkipRec-RL. As illustrated in Figure 5, we observe that (1) SkipRec-Gumbel and SkipRec-RL (i.e., the no pre-training versions) prevent overfitting better than the NextItNet+ on both Weishi (left) and ML20 (right); (2) NextItNet+ converges faster than SkipRec given that more parameters are updated in each round of backpropagation. For instance, on Weishi, the MRR@5 of NextItNet+ starts to decrease sharply after 4 epochs, whereas SkipRec (including the pre-trained versions) keeps relatively stable after convergence. These results are basically in accordance with the model structures and theoretical expectations.

5.4 Adaptability Experiment

To verify the generality of SkipRec, we specify it with SASRec (Kang and McAuley 2018) and report results in Table 5. Similar to NextItNet+, we also add a constant coefficient to the residual mappings, so that the model can stack even more

Model	Weishi					ML20					ML100				
	MRR@5	MRR@20	HR@5	HR@20	Speedup	MRR@5	MRR@20	HR@5	HR@20	Speedup	MRR@5	MRR@20	HR@5	HR@20	Speedup
MostPop	0.0050	0.0121	0.0187	0.0940	\	0.0044	0.0076	0.0134	0.0485	\	0.0040	0.0068	0.0124	0.0433	\
NFM	0.0734	0.0876	0.1307	0.2774	\	0.0483	0.0585	0.0856	0.1933	\	0.0341	0.0402	0.0573	0.1221	\
GRU4Rec	0.1001	0.1148	0.1649	0.3216	\	0.0938	0.1082	0.1577	0.3115	\	0.0973	0.1120	0.1611	0.3117	\
Caser	0.0911	0.1051	0.1498	0.2973	\	0.0915	0.1045	0.1509	0.2857	\	0.0927	0.1058	0.1505	0.2858	\
NextItNet	0.1025	0.1175	0.1669	0.3234	\	0.1019	0.1173	0.1686	0.3285	\	0.1073	0.1226	0.1752	0.3339	\
NextItNet+	0.1091	0.1243	0.1767	0.3354	1.00×	0.1067	0.1234	0.1784	0.3393	1.00×	0.1117	0.1276	0.1819	0.3465	1.00×
SkipRec-Gumbel	0.1105	0.1264	0.1796	0.3442	1.66×	0.1091	0.1249	0.1783	0.3424	1.92×	0.1122	0.1281	0.1827	0.3474	1.35×
Gumbel w/ pre	**0.1121**	**0.1279**	**0.1826**	0.3453	1.74×	**0.1104**	**0.1261**	**0.1800**	**0.3431**	1.72×	**0.1147**	**0.1306**	**0.1862**	**0.3511**	1.30×
SkipRec-RL	0.1098	0.1255	0.1788	0.3426	2.00×	0.1087	0.1243	0.1773	0.3402	2.28×	0.1111	0.1270	0.1799	0.3455	2.08×
RL w/ pre	0.1117	0.1275	0.1813	**0.3460**	2.01×	0.1101	0.1258	0.1793	0.3413	2.30×	0.1136	0.1292	0.1852	0.3488	2.03×

Table 2: Performance comparison on Weishi, ML20 and ML100. MostPop returns top-N items ranked by popularity.

Data	Model	MRR@5	MRR@20	HR@5	HR@20
Weishi	NextItNet+	0.1091	0.1243	0.1767	0.3354
	SkipRec-Gumbel	**0.1121**	**0.1279**	**0.1826**	0.3453
	SkipRec-RL	0.1117	0.1275	0.1813	**0.3460**
	Random policy	0.1056	0.1212	0.1727	0.3346
ML20	NextItNet+	0.1067	0.1234	0.1784	0.3393
	Skip-Gumbel	**0.1104**	**0.1261**	**0.1800**	**0.3431**
	SkipRec-RL	0.1101	0.1258	0.1793	0.3413
	Random policy	0.1018	0.1172	0.1676	0.3277

Table 3: Results with a random policy on Weishi and ML20.

Data	Model	MRR@5	MRR@20	HR@5	HR@20
Weishi	NextItNet+	0.1091	0.1243	0.1767	0.3354
	SkipRec-Gumbel	0.1121	0.1279	**0.1826**	0.3453
	SkipRec-Gumbel-GRU	**0.1123**	**0.1284**	0.1816	**0.3473**
	SkipRec-RL	0.1117	0.1275	0.1813	0.3460
	SkipRec-RL-GRU	0.1112	0.1267	0.1801	0.3413
ML20	NextItNet+	0.1067	0.1234	0.1784	0.3393
	SkipRec-Gumbel	**0.1104**	**0.1261**	0.1800	**0.3431**
	SkipRec-Gumbel-GRU	0.1100	0.1255	**0.1818**	0.3426
	SkipRec-RL	0.1101	0.1258	0.1793	0.3413
	SkipRec-RL-GRU	0.1096	0.1254	0.1780	0.3414

Table 4: Results with the policy network using CNN and GRU on Weishi and ML20.

than 100 layers (i.e.,128 layers for both Weishi and ML20)[4], and get the optimal results. We term it as SASRec+. Regarding the policy network, we implement it with a lightweight multi-head self-attention network, i.e., one standard attention residual block.

As shown, similar conclusions can be made as discussed in Section 5.1. SkipRec with adaptive policies yields competitive recommendations compared with the original SASRec+. Particularly, we observe that SkipRec with the Gumbel strategy and pre-training obtains around 2.4% improvements on Weishi. As analyzed before, the main benefits of SkipRec come from the attention policy, which is as powerful as the convolutional and recurrent networks.

[4]Note it is not comparable with NextItNet+ since it uses only 64 layers for experiments.

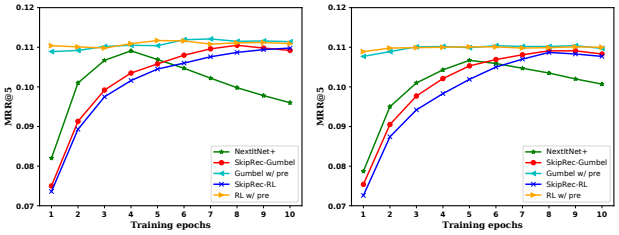


Figure 5: Convergence behaviors,w.r.t. MRR@5, on Weishi (left) and ML20 (right).

Data	Model	MRR@5	MRR@20	HR@5	HR@20	Speedup
Weishi	SASRec+	0.1076	0.1225	0.1749	0.3298	1.00×
	SkipRec-Gumbel	0.1089	0.1246	0.1770	0.3397	1.52×
	Gumbel w/ pre	**0.1102**	**0.1258**	**0.1778**	**0.3411**	1.37×
	SkipRec-RL	0.1086	0.1240	0.1761	0.3384	1.46×
	RL w/ pre	0.1094	0.1250	0.1768	0.3388	1.58×
ML20	SASRec+	0.1154	0.1310	0.1890	0.3498	1.00×
	SkipRec-Gumbel	0.1151	0.1307	0.1872	0.3500	1.57×
	Gumbel w/ pre	**0.1189**	**0.1346**	**0.1910**	**0.3538**	1.31×
	SkipRec-RL	0.1147	0.1304	0.1861	0.3493	1.69×
	RL w/ pre	0.1175	0.1333	0.1879	0.3524	1.53×

Table 5: Results by applying SASRec+ for SkipRec.

6 Conclusion and Future Work

In this paper, we have proposed an adaptive layer selection framework (SkipRec), whereby the number of network layers can be selected on a per-user basis. We devise a policy network to automatically determine which layers should be retained and which layers should be skipped in the backbone network. We also demonstrate that deep recommender models can be stacked with more than 100 layers and thereby it is necessary to come up strategies for efficient inference. We expect our studies will inspire new research in exploring deep, effective and efficient recommender models.

A small drawback of SkipRec is that it converges slightly slower than the backbone network due to fewer parameters are trained in each round. In the future, we would explore advanced techniques to speedup the training of such deep & large recommender models (such as in (Wang et al. 2020)) so as to free up more computational resources.

References

Bachlechner, T.; Majumder, B. P.; Mao, H. H.; Cottrell, G. W.; and McAuley, J. 2020. Rezero is all you need: Fast convergence at large depth. *arXiv preprint arXiv:2003.04887* .

Bengio, Y. 2013. Deep Learning of Representations: Looking Forward. In *SLSP*, Lecture Notes in Computer Science, 1–37. Springer.

Gu, Y.; Lei, T.; Barzilay, R.; and Jaakkola, T. S. 2016. Learning to refine text based recommendations. In *EMNLP*, 2103–2108.

Guo, Y.; Shi, H.; Kumar, A.; Grauman, K.; Rosing, T.; and Feris, R. S. 2019. SpotTune: Transfer Learning Through Adaptive Fine-Tuning. In *CVPR*, 4805–4814.

He, X.; and Chua, T. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*, 355–364. ACM.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Kang, W.; and McAuley, J. J. 2018. Self-Attentive Sequential Recommendation. In *ICDM*, 197–206.

Maddison, C. J.; Mnih, A.; and Teh, Y. W. 2017. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In Bilmes, J. A.; and Ng, A. Y., eds., *UAI*, 452–461. AUAI Press.

Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, 811–820. ACM.

Rennie, S. J.; Marcheret, E.; Mroueh, Y.; Ross, J.; and Goel, V. 2017. Self-Critical Sequence Training for Image Captioning. In *CVPR*, 1179–1195.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 1265–1295.

Smirnova, E.; and Vasile, F. 2017. Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks. In *DLRS@RecSys*, 2–9. ACM.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*, 1441–1450. ACM.

Sun, Y.; Yuan, F.; Yang, M.; Wei, G.; Zhao, Z.; and Liu, D. 2020. A Generic Network Compression Framework for Sequential Recommender Systems. *SIGIR* .

Tan, Y. K.; Xu, X.; and Liu, Y. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS@RecSys*, 17–22. ACM.

Tang, J.; and Wang, K. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*, 565–573. ACM.

Wang, J.; Yuan, F.; Chen, J.; Wu, Q.; Li, C.; Yang, M.; Sun, Y.; and Zhang, G. 2020. StackRec: Efficient Training of Very Deep Sequential Recommender Models by Layer Stacking. *arXiv preprint arXiv:2012.07598* .

Wang, P.; Guo, J.; Lan, Y.; Xu, J.; Wan, S.; and Cheng, X. 2015. Learning hierarchical representation model for nextbasket recommendation. In *SIGIR*, 403–412. ACM.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. E. 2018. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 409–424.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 229–256.

Wu, C.-Y.; Ahmed, A.; Beutel, A.; Smola, A. J.; and Jing, H. 2017. Recurrent recommender networks. In *WSDM*. ACM.

Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L. S.; Grauman, K.; and Feris, R. S. 2018. BlockDrop: Dynamic Inference Paths in Residual Networks. In *CVPR*, 8817–8826.

Ying, H.; Zhuang, F.; Zhang, F.; Liu, Y.; Xu, G.; Xie, X.; Xiong, H.; and Wu, J. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In *IJCAI*, 3926–3932.

Yu, F.; Liu, Q.; Wu, S.; Wang, L.; and Tan, T. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In Perego, R.; Sebastiani, F.; Aslam, J. A.; Ruthven, I.; and Zobel, J., eds., *SIGIR*, 729–732. ACM.

Yuan, F.; He, X.; Jiang, H.; Guo, G.; Xiong, J.; Xu, Z.; and Xiong, Y. 2020a. Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation. In *Proceedings of The Web Conference 2020*, 303–313.

Yuan, F.; He, X.; Karatzoglou, A.; and Zhang, L. 2020b. Parameter-Efficient Transfer from Sequential Behaviors for User Modeling and Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1469–1478.

Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J. M.; and He, X. 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.

Yuan, F.; Zhang, G.; Karatzoglou, A.; He, X.; Jose, J.; Kong, B.; and Li, Y. 2020c. One Person, One Model, One World: Learning Continual User Representation without Forgetting. *arXiv preprint arXiv:2009.13724* .