Cost-Effective On-Device Sequential Recommendation with Spiking Neural Networks

Di Yu[1], **Changze Lv**[2], **Xin Du**[1*], **Linshan Jiang**[3], **Qing Yin**[4], **Wentao Tong**[1], **Xiaoqing Zheng**[2] and **Shuiguang Deng**[1]
[1]Zhejiang University
[2]Fudan University
[3]National University of Singapore
[4]JD.com
yudi2023@zju.edu.cn, xindu@zju.edu.cn, dengsg@zju.edu.cn

Abstract

On-device sequential recommendation (SR) systems are designed to make local inferences using real-time features, thereby alleviating the communication burden on server-based recommenders when handling concurrent requests from millions of users. However, the resource constraints of edge devices, including limited memory and computational capacity, pose significant challenges to deploying efficient SR models. Inspired by the energy-efficient and sparse computing properties of deep Spiking Neural Networks (SNNs), we propose a cost-effective on-device SR model named SSR, which encodes dense embedding representations into sparse spike-wise representations and integrates novel spiking filter modules to extract temporal patterns and critical features from item sequences, optimizing computational and memory efficiency without sacrificing recommendation accuracy. Extensive experiments on real-world datasets demonstrate the superiority of SSR. Compared to other SR baselines, SSR achieves comparable recommendation performance while reducing energy consumption by an average of 59.43%. In addition, SSR significantly lowers memory usage, making it particularly well-suited for deployment on resource-constrained edge devices.

1 Introduction

Sequential recommendation (SR) systems [Wang *et al.*, 2019] have become indispensable to ubiquitous web applications such as Shopee and TikTok, owing to their exceptional ability to model users' historical behaviors and uncover latent preferences across a wide range of products and services, ultimately driving enhanced business revenue. Most conventional SR systems are entirely server-side solutions [Kang and McAuley, 2018; Lai *et al.*, 2024], heavily dependent on the cloud's substantial storage, memory, and computational resources [Steck, 2019]. However, this cloud-based paradigm faces critical challenges such as high reliance on network conditions, transmission latency, and significant risks to user privacy [Yao *et al.*, 2022; Xia *et al.*, 2024].

With the growing computational capabilities of intelligent edge devices, deploying SR models on the edge becomes a promising solution. By enabling local inference, edge-based SR systems reduce communication overhead, improve real-time responsiveness, and provide inherent privacy advantages, as sensitive data remains local to the device [Xia *et al.*, 2022; Yin *et al.*, 2024]. However, current on-device SR models face potential challenges like:

High Memory Overheads. In standard SR tasks, item token sequences are used as the sole modality, requiring high-dimensional embeddings to capture generalized representations for each item [Zivic *et al.*, 2024]. This results in substantial memory footprints, particularly for platforms with extensive item catalogs. For instance, Amazon must manage over 350 million product embeddings for its recommendation system [Chen *et al.*, 2022], leading to inefficient resource utilization and increased operational costs.

High Energy Costs. On-device SR models often operate on battery-powered mobile devices, such as smartphones, where sustainability is critical. Current SR models rely on artificial neural networks (ANNs) with large-scale architectures and full-precision computations, resulting in significant energy consumption. For example, deploying ANN-based models on devices like Google Glass [Venkataramani *et al.*, 2016] consumes 0.15J per frame for video streams, which restricts the battery life of 2.1WH capacity to just 25 minutes, thereby degrading usability and user experience.

To address these challenges, we propose a **S**pike-wise **S**equential **R**ecommendation (SSR) model[1], leveraging the computational paradigm of spiking neural networks (SNNs). This model is well-suited for deployment on intelligent edge devices featuring neuromorphic chips [Yao *et al.*, 2024], which are specialized hardware designed to mimic the neural network architecture for efficient, low-energy computation. To overcome the limitations of dense embeddings in memory and computation, we introduce a neural encoding method that converts the original embedding table into spike-

[*] Corresponding Author: Xin Du and Shuiguang Deng.

[1] https://github.com/AmazingDD/serenRec

based representations, allowing SSR to process sequential features more efficiently. This method can diminish the storage limitations and accelerate computation compared to conventional dense embedding vectors. In addition, recent breakthroughs of deep SNNs in sequential tasks [Lv *et al.*, 2024; Xing *et al.*, 2024] shed light on the feasibility of developing on-device SR models due to their advantages of energy efficiency [Maass, 1997]. This study includes spiking filter modules that emulate the Fourier transform mechanism, enabling SSR to effectively process spike-wise signals and extract the dynamic user preferences from the asynchronous discrete spike trains.

To demonstrate the effectiveness of SSR, we conduct extensive experiments across real-world datasets. The results show that SSR achieves comparable or even better recommendation performance with much less computational energy cost compared to multiple ANN-based state-of-the-art (SOTA) baselines. SSR also generates sparse item representations to facilitate calculation, reduce memory footprint, and speed up inference, which is essential for on-device SR deployment [Yin *et al.*, 2024]. To conclude, our contributions can be summarized as follows:

- This study first proposes an on-device SR model with deep SNNs, offering a cost-effective and energy-efficient solution for enhancing user experience on intelligent edge devices.
- SSR generates a binary item representation, significantly reducing memory usage. It also incorporates spiking filter modules, effectively extracting individual preferences from asynchronous and sparse inputs.
- Extensive experimental results on **five** datasets validate that SSR can achieve recommendation performance on par with or even superior to **seven** SOTA SR models with much less energy consumption.

2 Related Work

On-Device Sequential Recommendation. Recently, on-device SR systems have emerged to address users' demands for low-latency responses and heightened data privacy concerns [Yin *et al.*, 2024]. Aerorec [Xia *et al.*, 2024] introduces a self-supervised knowledge distillation method to mitigate the accuracy loss in compressed on-device recommendation models. OD-Rec [Xia *et al.*, 2023] compresses the item embedding table by discrete compositional code learning to implement on-device recommendation efficiently. DIET [Fu *et al.*, 2024] tailors SR models for each device to minimize bandwidth usage and storage consumption. Existing methods typically reduce inference latency and energy consumption through techniques such as compression [Yuan *et al.*, 2020] and quantization [Li *et al.*, 2023]. However, these approaches often degrade model performance by reducing feature expressiveness, particularly in scenarios with large-scale recommendation systems [Hou *et al.*, 2023]. This motivates us to explore a novel on-device SR model that balances computational energy cost and recommendation accuracy better.

Recommendation with Spiking Neural Networks. Several researchers have investigated the feasibility of integrating SNNs into conventional collaborative filtering (CF) recommendation models. [Zhu *et al.*, 2022] introduces spike-wise graphs to reduce the memory cost in graph-based CF models. [Ma *et al.*, 2023; Agarwal *et al.*, 2024] utilize the sparse encoding mechanism of SNNs to convert raw features into spike signals, thereby improving the CF model performance. These studies demonstrate the potential of SNNs to enhance feature representation and improve recommendation performance. However, most existing works focus on CF models and neglect SR tasks. Furthermore, no SNN-related SR algorithms have been designed specifically for neuromorphic chips [Ma *et al.*, 2024], leaving SNNs' low-energy advantages under-explored for energy-aware edge device deployment. To bridge this gap, we propose SSR, the first on-device SR model utilizing spiking neural networks to achieve energy-efficient sequential recommendation.

3 Preliminary

3.1 Spiking Neural Network

Leaky Integrate-and-Fire (LIF) [Maass, 1997] is one conventional spiking neuron to build SNNs, whose calculations can be described by a series of discrete-time equations:

$$U(t) = (1-\tau)\cdot H(t-1) + \tau \cdot I(t) \tag{1}$$

$$O(t) = \Theta(U(t) - \overline{V}) \tag{2}$$

$$H(t) = U(t)\cdot(1-O(t)) + V_r \cdot O(t) \tag{3}$$

where $\tau, \overline{V}$, and V_r denote the decay factor, threshold, and reset potential. $\Theta(\cdot)$ is the non-differentiable Heaviside step function determining whether a spike is generated at time step t according to the membrane potential $U(t)$ updated by input stimulus $I(t)$. In this study, we use $\mathcal{SN}(\cdot)$ to one LIF layer for brevity and introduce BPTT with surrogate functions [Dampfhoffer *et al.*, 2023] to train SNNs on GPUs.

3.2 Fourier Transform

Discrete Fourier transform (DFT) has demonstrated its effectiveness in SR applications [Zhou *et al.*, 2022; Shin *et al.*, 2024]. As the input data of SR is one-dimensional sequences $\{x_n\}_{n=1}^{N}$ with the length of N, we only consider 1D-DFT in our study, which converts the original sequence into a sequence of complex numbers in the frequency domain by:

$$z_k = \sum_{n=1}^{N} x_n e^{-\frac{2\pi \mathrm{i}}{N}nk} = \sum_{n=1}^{N} x_n W^{nk}, 1 \le k \le N \tag{4}$$

where $W \equiv e^{\frac{-2\pi \mathrm{i}}{N}}$ is a complex number as the twiddle factor, z_k represents the signal with frequency $2\pi k/N$. Through Equation (4), values in the sequence are decomposed into components of different frequencies. Since DFT is a one-to-one mapping operation in the time and frequency domains, the frequency representation $\{z_k\}_{k=1}^{N}$ can be converted to the original feature domain by inverse DFT (IDFT):

$$x_n = \frac{1}{N}\sum_{k=1}^{N} z_k W^{-nk} \tag{5}$$

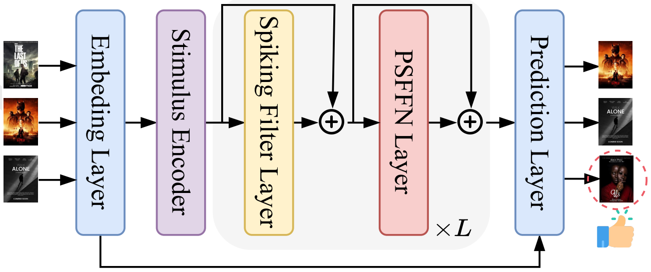

Figure 1: Overview of SSR.

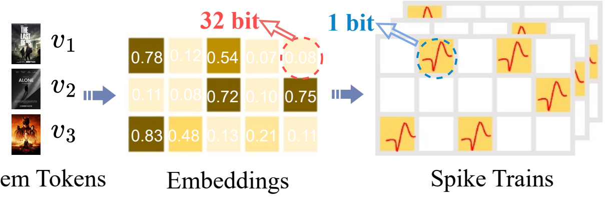

Figure 2: Comparison between conventional and spike-wise token representations. Instead of full-precision (*32-bit*) embeddings, spike trains provide sparser and more compact *1-bit* binary representations for resource-constrained edge devices to compute efficiently.

Since item sequences can be treated as users' periodic behavior mixed with some unexpected actions (noise), DFT offers an efficient method to capture these time features [Du *et al.*, 2023]. This study uses the Fourier transform paradigm to process and filter noisy spikes in SSR. We denote DFT and IDFT by $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot)$, respectively.

4 Methodology

4.1 Problem Statement

We first define $\mathcal{U}$ and $\mathcal{V}$ to represent the set of users and items, respectively, where $u \in \mathcal{U}$ denotes a user and $v \in \mathcal{V}$ denotes an item. The number of users and items are $|\mathcal{U}|$ and $|\mathcal{V}|$, respectively. Sequential recommendation relies on the previous interactions in a short period to generate the next-item recommendation. Let $\mathbf{x} = \{v_1, v_2, ..., v_N\} \in \mathcal{X}$, where N is the number of interactions, and v_n is the n-th item that the user u^2 has interacted with in the chronological order. For convenience, we use $\mathbf{x}_{j:k}$ to denote the subsequence, i.e., $\mathbf{x}_{j:k} = \{v_j, ..., v_k\}$, where $1 \leq j \leq k \leq N$.

Based on the above notations, we define the task of the SR as to predict the next item v_{N+1} that the user is most likely to interact with at the $(N+1)$-th step, given the contextual item sequences $\mathbf{x} = \{v_1, v_2, ..., v_N\}$ of a user u. We set y as the ground truth of the following user action, i.e., $y = v_{N+1} \in \mathcal{Y}$. Hence, the learning objective of implementing SR on resource-constrained edge devices is to minimize the ranking loss $\mathcal{L}$ over data distribution $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$:

$$\begin{aligned} &\min_{\theta} \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \mathcal{L}(f(\mathbf{x};\theta), y) \\ &\text{s.t. } \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}} \mathcal{C}(f(\mathbf{x};\theta)) \leq b \end{aligned} \tag{6}$$

where $\mathcal{C}(\cdot)$ refers to the total cost of using a specific SR model for inference, e.g., energy consumption, inference latency, etc., and b is the specific limitations required by the devices.

4.2 SSR Model

As depicted in Figure 1, we introduce the architecture of SSR, including five modules: embedding layer, stimulus encoder, spiking filter layer, PSFFN layer, and prediction layer.

Embedding Layer. Similar to previous study [Yue *et al.*, 2024], an item embedding matrix $\mathbf{E}_V \in \mathbb{R}^{|\mathcal{V}|\times D}$ is leveraged to project each item token v_n in the sequence $\mathbf{x}$ to a high-dimensional representation $\mathbf{v}_n \in \mathbb{R}^D$. Given one item sequence $\mathbf{x}$, we can obtain a new sequence embedding matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ with look-up operations:

$$\mathbf{X} = \mathbf{E}_V(\mathbf{x}) \tag{7}$$

Although using non-spiking values like Equation (7) as inputs of SNN-based models is pretty ordinary [Yao *et al.*, 2023], it is more logical to utilize spikes as the input of SNNs like the brain trades in the global currency of the spike, bringing potential computational efficiency and sparse storage.

As depicted in Figure 2, one dense embedding table can be converted into T sparse binary tables by the temporal encoding mechanism of spiking neurons (In this study, N should always be ensured to be divisible by T). Due to the high sparsity and huge memory usage gap between floating-point and binary values (e.g., 32-bit vs. 1-bit), encoding the original embedding table with spiking neurons can enhance computational efficiency and diminish redundant memory usage. Therefore, we feed features from the embedding table, like a gray-scale image, to a layer of spiking neurons repeatedly over T time steps. Then, we get the spike-wise sequence embeddings $\mathbf{X}^* \in \mathbb{I}^{T\times N\times D}$ containing only binary values:

$$\mathbf{X}^* = \mathcal{SN}(\mathbf{E}_V(\mathbf{x})) \tag{8}$$

where T, N, and D are the degree of time step, sequence length, and hidden factor dimension.

Stimulus Encoder. The raw spike trains derived from Equation (8) offer a sparse representation of each item token but fail to capture sequential features. To address this limitation, we propose an encoder module within SSR, designed to aggregate spike-level signals and generate stimulus representations encompassing sequential features. This module comprises two components: (1) The serial presentation encoder is implemented as a linear transformation moving block. It captures cumulative spike information across temporal steps in the sequence, effectively extracting long-term interests. (2) The parallel presentation encoder acts as a global-level aggregator, leveraging another weight matrix to perform weighted summations over items in the sequence at each time step, thereby capturing point-level preference patterns. The stimuli generated by these two encoders are aggregated to form

[2] When the user is anonymous, sequential recommendation tasks degenerate into session-based recommendation tasks.

a new dense stimulus that preserves the sequences' temporal and contextual patterns, which will then be fed into a LIF layer $\mathcal{SN}(\cdot)$ to construct the spike-wise encoded sequential features $\mathbf{H} \in \mathbb{I}^{T\times N\times D}$ for further model learning. Considering the primary focus of this study is designing a feasible SNN-based model to complete on-device SR tasks, we have proved that our stimulus encoding method fits well in most scenarios.

Spiking Filter Layer. Inspired by [Zhou *et al.*, 2022], filtering operation can be utilized to convert each dimension of features into another learnable feature space and reconstruct the sequence representations with less noisy signals.

Given the input $\mathbf{H}_t^l \in \mathbb{I}^{N\times D}$ for the filter layer in the l-th block at time step t ($l \in [1, L]$, $t \in [1, T]$; when l=1, $\mathbf{H}^l$=$\mathbf{H}$), we apply 1D-DFT $\mathcal{F}(\cdot)$ along the sequence dimension N of $\mathbf{H}^l$ to convert it to frequency domain and then multiply a learnable matrix $\mathbf{W}^l$, denoted as:

$$\tilde{\mathbf{H}}_t^l = \mathcal{F}(\mathbf{H}_t^l) \odot \mathbf{W}^l \tag{9}$$

where $\odot$ is the element-wise multiplication. $\tilde{\mathbf{H}}_t^l$ can be adaptively optimized using gradient descent to represent an arbitrary filter in the frequency domain. Finally, the 1D-IDFT $\mathcal{F}^{-1}(\cdot)$ is adopted to transform the modulated spectrum $\tilde{\mathbf{H}}_t^l$ back to the original domain, reconstructing new sequential representations as the output of the entire filter layer:

$$\mathbf{F}_t^l = \mathcal{F}^{-1}(\tilde{\mathbf{H}}_t^l) \tag{10}$$

These operations allow useless spikes from $\mathbf{H}_t^l$ to be effectively filtered as noise. Following [Kang and McAuley, 2018]. We add the skip connection [Wang *et al.*, 2021] to avoid gradient vanishing and unstable training issues.

However, the above operations are feasible for ANNs on GPUs yet unsuitable for SNNs on neuromorphic chips since $\tilde{\mathbf{H}}_t^l$ is a complex tensor that is an illegal representation in neuromorphic computing logistics. To tackle this problem, we exploit the successive multiplication representation, emulating DFT in SNN layers due to DFT's linear transformation [Arsalan *et al.*, 2023]. We first rewrite Equation (4) according to the Euler's method as:

$$z_k = \sum_{n=1}^{N} x_n \left(\left[\cos(\frac{2\pi}{N}nk) \right] - \left[\sin(\frac{2\pi}{N}nk) \right] \mathrm{i} \right) \tag{11}$$

where n and k take values ranging from 1 to N. When expanding x_n to an input vector, Equation (11) can be written in matrix form like:

$$\mathbf{Z} = (\mathbf{W}_R\mathbf{X}) + (\mathbf{W}_I\mathbf{X})\mathrm{i} \tag{12}$$

where $\mathbf{W}_R$ and $\mathbf{W}_I$ are the real and imaginary coefficient matrices, whose individual coefficient values can be calculated using the following equations:

$$\mathbf{W}_R[n, k] = \cos(\frac{2\pi}{N}nk) \tag{13}$$

$$\mathbf{W}_I[n, k] = -\sin(\frac{2\pi}{N}nk) \tag{14}$$

Algorithm 1 Model Inference of SSR.

```
Input: Input sequences $\mathbf{x}$, top value $k$
Output: Top-$k$ recommendation list $P_k$
Query spike-wise representation $\mathbf{X}^*$ with $\mathbf{x}$.
Encoding $\mathbf{X}^*$ to $\mathbf{H}$.
for all filter block $l \in \{1, ..., L\}$ do
    Learnable 1D-DFT: $\tilde{\mathbf{H}}_l = \mathcal{F}(\mathbf{H}^l) \odot \mathbf{W}^l$.
    1D-IDFT: $\tilde{\mathbf{F}}_l = \mathcal{F}^{-1}(\tilde{\mathbf{H}}_l)$.
    Convert $\tilde{\mathbf{F}}_l$ to spikes with $\mathcal{SN}(\cdot)$.
end for
Densify: $\hat{\mathbf{H}}^L \leftarrow \mathrm{Linear}(\tilde{\mathbf{F}}_L)$.
Computing preference scores $P$ with $\hat{\mathbf{H}}^L$ and $\mathbf{X}^*$.
Sort $P$ in a descending order.
Cut out top-$k$ items from $P$ to form $P_k$.
return $P_k$
```

We can then substitute $\mathcal{F}(\mathbf{H}_t^l)$ in Equation (9) with a series of spike-wise operations like:

$$\tilde{\mathcal{F}}(\mathbf{H}_t^l) \leftarrow \mathcal{SN}(\mathrm{Conv}(\mathcal{SN}([\mathbf{W}_R\mathbf{H}_t^l; \mathbf{W}_I\mathbf{H}_t^l]))) \tag{15}$$

where the convolution operation Conv simulates the integration of real and imaginary parts as the complex tensor. Similarly, the IDFT operation in Equation (10) can also be modified to spike-wise form inversely, denoted as $\tilde{\mathcal{F}}^{-1}(\cdot)$, and we denote the binary output of the l-th filter layer after implementing the spike-wise IDFT operation as $\tilde{\mathbf{F}}_t^l$.

PSFFN Layer. For the point-wise spiking feed-forward network (a.k.a. PSFFN) layer in the l-th block at time step t, we integrate LIF with a two-layer MLP to project the nonlinear features extracted from the spiking filter layer into a higher-dimensional space to improve the modeling ability of user actions in the hidden dimension. The computation of PSFFN is defined as:

$$\hat{\mathbf{H}}^l = \mathcal{SN}(\mathbf{W}_2\mathcal{SN}(\mathbf{W}_1\tilde{\mathbf{F}}_t^l)) + \tilde{\mathbf{F}}_t^l \tag{16}$$

where $\mathbf{W}_1, \mathbf{W}_2$ are both learnable weight matrices. Skip connection operations are also incorporated into this component for training stability.

Prediction Layer. Given the spike-wise hidden features $\hat{\mathbf{H}}_t^L$ at time step t from the last L-th block at position n, it will first be converted into a dense vector as $\hat{h}_{t,n}^L \in \mathbb{R}^D$ via linear transformation in the prediction layer. Then, the preference score for the next item y over $\mathcal{V}$ given sequence $\mathbf{x}$ can be calculated by:

$$P(y = v_n|\mathbf{x}_{1:n-1}) = \sum_{t=1}^{T} \mathbf{X}_t^*(v)\hat{h}_{t,n}^L + b_v^o, v_n \in \mathcal{V} \tag{17}$$

where $b^o \in \mathbb{R}^{|\mathcal{V}|}$ is an additional bias term indicating the deviation of items $v \in \mathcal{V}$ from the global perspective. We adopt point-wise cross-entropy loss to optimize the model parameters θ by minimizing the following objective:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\log(P(y = v_n|\mathbf{x}_{1:n-1})) \tag{18}$$

We summarize the inference process of SSR in Algorithm 1. In addition, we have elaborated on the time complexity analysis of SSR and the effectiveness of the spiking filter module .

Dataset	#Sequences	#Items	#Inter.	Density (↓)
Movielens	993,571	3,416	999,611	4.84×10^{-2}
Steam	982,712	8,606	1,096,673	1.18×10^{-3}
Music	150,690	11,269	166,942	9.12×10^{-4}
Video	441,760	17,286	496,904	5.21×10^{-4}
Arts	436,614	22,611	492,583	3.89×10^{-4}

Table 1: Statistics of the experimental datasets after preprocessing. "*#Inter.*" represents the total user-item interactions. The maximum sequence length is set to 20 as default in this study.

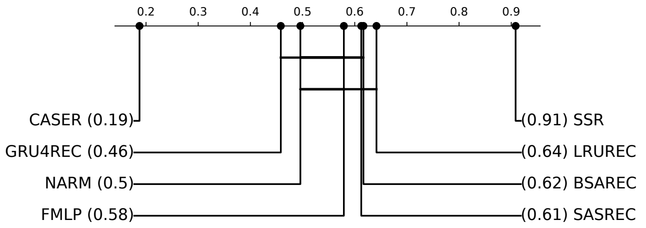

Figure 3: Critical Difference (CD) diagram of all models in Table 2 with a confidence level of 95%.

5 Experiments

5.1 Experimental Settings

Datasets. Five public datasets collected from various platforms are selected to evaluate the efficacy of SSR. Their corresponding statistics are summarized in Table 1, sorted by density. Following [Zhou *et al.*, 2022], we group the records by users or sessions, sort them by time in ascending order, and adopt a 5-core strategy for all datasets to ensure each user and item has at least five interaction records.

Metrics. As for the recommendation performance, we follow [Yin *et al.*, 2023] and calculate the Hit Rate (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR) after generating the Top-k recommendation list with a full-ranking protocol. Higher values indicate better performance for all metrics. For energy cost, we evaluate the number of operations (OPs) [Xing *et al.*, 2024] and theoretical energy consumption gauged in joules (J).

Baselines. We compare SSR with multiple SOTA SR baselines with two categories: (1) **RNN-/CNN-based SR models**: *CASER* [Tang and Wang, 2018], *GRU4REC* [Hidasi *et al.*, 2015] and *NARM*; (2) **Transformer-based SR models**: *SASREC* [Kang and McAuley, 2018], *FMLP* [Zhou *et al.*, 2022], *BSAREC* [Shin *et al.*, 2024], and *LRUREC* [Yue *et al.*, 2024].

Implementation Details. We use the time-aware and user-level split-by-ratio strategy [Sun *et al.*, 2022] to split the whole dataset, where the last 20% of the total item sequences is the test set. To make a fair comparison, all model hyperparameters are searched with the TPE [Bergstra *et al.*, 2011] strategy within 20 trials with Optuna [Akiba *et al.*, 2019]. By default, we implement all models with Torch and use Adam optimizer with the 10^{-3} learning rate. The embedding dimension for each item is 64, and the time step size T for LIF neurons is 4.

5.2 Performance Evaluation

Recommendation Performance. Table 2 lists the results of our proposed SSR compared with all other baselines on various evaluation metrics across different datasets. We conclude from the observations that SSR achieves **comparable** performance across all metrics and datasets, with an average performance gap of 1.06% compared to the best method. Despite the data flow inside SSR being binary spikes, SSR maintains the effectiveness of capturing useful sequence characteristics, just as the floating-point operation does. Notably, SSR's performance gains are more significant on dense datasets. In contrast, they are modest on sparse datasets since dense sequences contain more sequential embedding knowledge fed into LIF neurons, thereby generating more spike information to communicate among layers and achieve better performance. Besides, SSR often performs better in ranking-related metrics. For instance, although BSAREC achieves a higher average hit rate, approximately 2.95% greater than SSR on the Steam and Arts datasets, SSR outperforms BSAREC on the NDCG metric by about 2.28%. This suggests that, while SSR may not always accurately identify all ground truth next items from users' previous interactions, it better captures sequential patterns, placing the ground truth items in higher positions within the ranking list. Meanwhile, by examining the critical difference diagram depicted in Figure 3, we can roughly categorize all the methods we have employed into three tiers, and it is a pleasant surprise that our SSR ranks with the top tier, demonstrating that regardless of the data scenario, SSR consistently demonstrates strong and stable performance, making it less likely to yield poor results when used for recommendations.

Energy Cost. We evaluate the inference energy efficiency to illustrate the potential of SSR to enhance user experience by prolonging mobile phone usage time while maintaining effective recommendation performance. We assume running SSR on a 45nm neuromorphic hardware [Horowitz, 2014] and other baselines on GPUs, since SNNs can demonstrate low computing energy costs when deployed on neuromorphic chips, and GPUs are the most suitable platform for executing ANNs. Since different baselines vary in computational complexity due to differences in parameters and architectures, directly comparing their energy consumption with SSR would be unfair. Hence, we devise an ANN-based variant called SSR-ANN, referring to SSR, which utilizes the standard DFT/IDFT computation, replaces all spiking neurons with ReLU[3], removes the stimulus encoder, and keeps other parameters the same as SSR's.

As listed in Table 3, SSR exhibits a notable energy cost reduction compared to its ANN variants, with an average decrease of approximately 59.43% across all datasets, and the computational operations are reduced by over 50%—such intriguing benefits position SNNs as an attractive choice for energy-efficient solutions for the on-device SR tasks[4].

[3] LIF neuron in SNN is an unbiased estimator of ReLU activation function over time [Rueckauer *et al.*, 2017].

[4] In practical applications, beyond computational costs, SSR executed on a neuromorphic chip can *eliminate the additional energy*

Methods	Metrics	GRU4REC	CASER	NARM	SASREC	FMLP	BSAREC	LRUREC	SSR
Movielens	HR@10	0.2120	0.2120	0.2205	0.2264	0.2140	0.2308	0.2090	**0.2312**
	MRR@10	0.0853	0.0880	0.0906	0.0907	0.0819	0.0928	0.0781	**0.0959**
	NDCG@10	0.1147	0.1168	0.1208	0.1222	0.1123	0.1264	0.1084	**0.1274**
Steam	HR@10	0.2020	0.1943	0.2031	0.2064	0.2020	0.2025	**0.2094**	0.2049
	MRR@10	0.1403	0.1321	0.1422	0.1451	0.1439	0.1418	**0.1461**	0.1446
	NDCG@10	0.1547	0.1466	0.1564	0.1594	0.1575	0.1559	**0.1609**	0.1586
Arts	HR@10	0.1590	0.1396	0.1560	0.1622	0.1638	**0.1661**	0.1606	0.1646
	MRR@10	0.1189	0.0995	0.1175	0.1177	0.1182	0.1130	0.1180	**0.1194**
	NDCG@10	0.1283	0.1090	0.1265	0.1282	0.1299	0.1256	0.1281	**0.1301**
Music	HR@10	0.1511	0.1161	0.1498	0.1650	0.1604	0.1725	**0.1788**	0.1684
	MRR@10	0.0895	0.0582	0.0947	0.0935	0.0947	0.0966	**0.0988**	0.0968
	NDCG@10	0.1040	0.0718	0.1078	0.1104	0.1104	0.1137	**0.1178**	0.1148
Video	HR@10	0.1297	0.1127	0.1293	0.1273	0.1331	0.1396	0.1334	**0.1424**
	MRR@10	0.0733	0.0582	0.0709	0.0696	0.0699	0.0664	0.0687	**0.0746**
	NDCG@10	0.0865	0.0709	0.0845	0.0830	0.0868	0.0835	0.0838	**0.0875**
T-Test (SSR, *)	*p*-value	0.6328	0.1789	0.7010	0.8236	0.7283	0.8893	0.8000	1.0000

Table 2: Performance comparison against the other baselines. **Bold** and underline values are the best and second-best results across all methods in one dataset with the best-searched hyperparameter settings. We run three trials and report the mean metric results. Pair-wise t-tests are implemented between SSR and each baseline to determine the significance of the entire performance difference.

Dataset	SSR		SSR-ANN	
	OPs (M)	Energy (mJ)	OPs (M)	Energy (mJ)
Movielens	11.27	0.51	26.60	1.22
Steam	4.97	0.23	12.22	0.56
Arts	6.22	0.28	13.18	0.67
Video	7.69	0.35	18.70	0.86
Music	11.95	0.53	27.72	1.27

Table 3: Computational efficiency across different datasets when making batch inferences. OPs refer to Synaptic Operations (SOPs) in SNNs and Floating Point Operations (FLOPs) in ANNs.

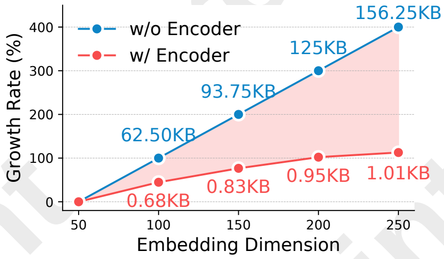

Figure 4: In-memory size per item token representation *w.r.t* dimension for conventional embedding and spike-wise embedding.

5.3 Efficiency Evaluation

Memory Usage. We vary the dimensionality of item embeddings from 50 to 250 to evaluate the compression effect of LIF neurons on the original dense embeddings. Observations in Figure 4 reveal that the parameter size of the conventional model increases linearly with the embedding dimension. In contrast, the embedding size regulated by the LIF layer encoder is substantially reduced, achieving a maximum reduction rate of 99.36%. Moreover, as the embedding dimension grows, the size follows a logarithmic growth pattern. This demonstrates that spike-wise embedding effectively compresses and retains essential information, reducing computational memory overheads for resource-constrained edge devices via sparse tensor computation.

Model Robustness. We conduct experiments by randomly replacing different percentages of items in sequences with randomly selected fake items and retraining the model using the corrupted sequences as input to evaluate the robustness of SSR against data noise issues. The noise ratios varied from 0% to 20%, with increments of 5%. SASREC and NARM are the target models for comparison, as they feature representative neural architectures within their respective baseline categories. We adopt relative metrics, i.e., Relative (Rel.) HR@10 and Rel. NDCG@10, to assess the performance trend of different models *w.r.t* different levels of noise perturbation. Figure 5 exhibits the performance degradation under different noise levels, indicating the potential of SNNs in addressing data noise issues. In the sparser Arts dataset, while SASREC and NARM initially perform well under low noise levels, their performance degrades significantly when exposed to 15% noise. In contrast, SSR demonstrates greater robustness to noise compared to these models. Even with 20% noise, SSR maintains a relative HR@10 of 98.92% and an NDCG@10 of 98.18%. In the relatively denser Movie-

consumption associated with ***data movement*** *between the chip and memory*. Furthermore, neuromorphic chips offer significantly lower latency—about 200 times less—than conventional chips. For instance, Speck [Yao *et al.*, 2024] achieves a latency of less than 0.1 ms when implementing SNNs on-chip. In contrast, the latency increases to 24.7 ms when the SNN is not implemented on-chip.

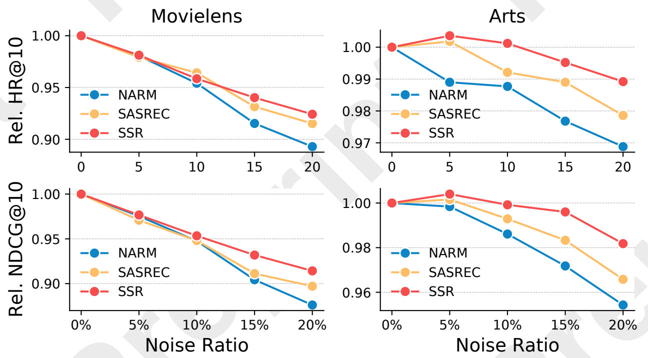

Figure 5: Relative (Rel.) metrics performance degradation *w.r.t* noise ratio across datasets with different densities.

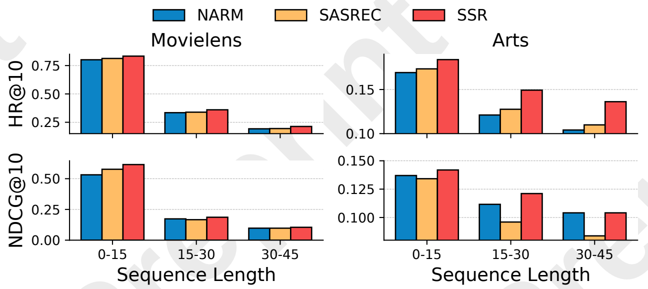

Figure 6: Model recommendation performance *w.r.t* sequence length across datasets with different densities.

lens dataset, SSR's performance is comparable to that of other models when the noise degree is less than 15% and then declines more slowly when noisier items are included. The superiority of SSR might be attributed to SNNs' advantage in capturing sparse features. When item sequences are sparse and have low similarity, introducing noise can easily lead to over-fitting in ANN-based models. However, SNNs' sparse computational characteristics can effectively mitigate this issue. When the sample is relatively dense and similar, the noise impact can be alleviated through knowledge transfer between samples, resulting in minimal performance differences among these models. Nevertheless, when the noise level is sufficiently high, causing a rapid decrease in sample similarity, the advantages of SNNs become more evident.

Long Sequence Scenarios. We evaluate the performance consistency of SSR across SR tasks with varying task complexities by altering the maximum length of the input sequences, as the high-order sequential dependencies introduced by long item sequences pose challenges for models to capture user preferences accurately. Similarly, we select SASREC and NARM as baselines to compare their performance with SSR and categorize sequences into three groups based on the sequence length. As illustrated in Figure 6, the models generally exhibit higher predictive accuracy for shorter input sequences, as leveraging more interaction histories can extract more valuable information through SR models. Meanwhile, SSR consistently demonstrates more excellent stability than the other models. In sparser Arts data scenarios, SSR outperforms other baselines, achieving a higher average HR@10 improvement rate than the dense Movielens scenario, with an approximate increase of 17.31%. This may be attributed to the spike-wise signals propagated within the SSR model, which are better at filtering out the items in an input sequence that are most relevant to the prediction. This reduces the impact of noise on the results, thereby more effectively uncovering users' latent preferences under long sequence conditions.

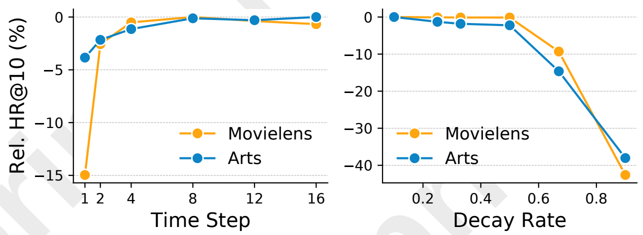

Figure 7: Sensitivity of SNN-related hyper-parameters in SSR.

5.4 Hyper-Parameter Analysis

Time Step. Time steps T dictate the complexity of SSR, and a higher number of time steps leads to an increased energy cost. As depicted in the left part of Figure 7, the HR@10 of SSR shows a notable improvement of approximately 15% when the number of time steps increases from 1 to 4. Beyond this point, the performance gains plateau, suggesting that the membrane potential of LIF neurons in SSR has stabilized and can generate consistent spike trains, with no additional time steps needed for stabilization. Hence, setting T=4 provides a practical solution for achieving a better cost-accuracy trade-off in SSR during inference.

Decay Rate. Decay rate τ determines the frequency of information communication between layers in SSR by controlling the firing rate of neurons. A lower decay rate indicates more spike transmission. The right part of Figure 7 shows that HR@10 remains relatively stable at lower decay rates but experiences a sharp decline of up to 40% once the decay rate exceeds 0.5, indicating that higher decay rates hinder the model's ability to capture relevant information, leading to the loss of crucial temporal details, which ultimately degrades HR performance. Meanwhile, lower decay rates often cause LIF neurons to fire spikes excessively, leading to unnecessary energy costs with minimal performance improvements. Therefore, a moderate decay rate like 0.5 can better balance the cost and performance of SSR.

6 Conclusion

In this paper, we proposed a novel on-device SR model called SSR, leveraging SNNs to achieve energy-efficient and sparse computing. SSR transforms dense item embeddings into sparse spike-wise representations, effectively capturing dynamic user preferences while reducing memory and computational overheads. Extensive experiments on diverse real-world datasets demonstrate that SSR achieves comparable recommendation accuracy to ANN-based baselines while significantly improving theoretical energy efficiency.

Acknowledgments

The work of this paper is supported by the National Key Research and Development Program of China under Grant 2022YFB4500100, the National Natural Science Foundation of China under Grant 62125206, the Zhejiang Provincial Natural Science Foundation of China under Grant No. LD24F020014, the National Key Research and Development Program of China No. 2024YDLN0005, and the Regional Innovation and Development Joint Fund of the National Natural Science Foundation of China No. U22A6001.

References

[Agarwal *et al.*, 2024] Gaurav Agarwal, Shail Kumar Dinkar, and Ajay Agarwal. Binarized spiking neural networks optimized with nomadic people optimization-based sentiment analysis for social product recommendation. *Knowledge and Information Systems*, 66(2):933–958, 2024.

[Akiba *et al.*, 2019] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[Arsalan *et al.*, 2023] Muhammad Arsalan, Avik Santra, and Vadim Issakov. Power-efficient gesture sensing for edge devices: mimicking fourier transforms with spiking neural networks. *Applied Intelligence*, 53(12):15147–15162, 2023.

[Bergstra *et al.*, 2011] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[Chen *et al.*, 2022] Yankai Chen, Huifeng Guo, Yingxue Zhang, Chen Ma, Ruiming Tang, Jingjie Li, and Irwin King. Learning binarized graph representations with multi-faceted quantization reinforcement for top-k recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 168–178, 2022.

[Dampfhoffer *et al.*, 2023] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Backpropagation-based learning techniques for deep spiking neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[Du *et al.*, 2023] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S Sheng. Frequency enhanced hybrid attention network for sequential recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 78–88, 2023.

[Fu *et al.*, 2024] Kairui Fu, Shengyu Zhang, Zheqi Lv, Jingyuan Chen, and Jiwei Li. Diet: Customized slimming for incompatible networks in sequential recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 816–826, 2024.

[Hidasi *et al.*, 2015] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[Horowitz, 2014] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014.

[Hou *et al.*, 2023] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171, 2023.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[Lai *et al.*, 2024] Riwei Lai, Rui Chen, Qilong Han, Chi Zhang, and Li Chen. Adaptive hardness negative sampling for collaborative filtering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8645–8652, 2024.

[Li *et al.*, 2023] Shiwei Li, Huifeng Guo, Lu Hou, Wei Zhang, Xing Tang, Ruiming Tang, Rui Zhang, and Ruixuan Li. Adaptive low-precision training for embeddings in click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4435–4443, 2023.

[Lv *et al.*, 2024] Changze Lv, Yansen Wang, Dongqi Han, Xiaoqing Zheng, Xuanjing Huang, and Dongsheng Li. Efficient and effective time-series forecasting with spiking neural networks. *arXiv preprint arXiv:2402.01533*, 2024.

[Ma *et al.*, 2023] Ying Ma, Gang Chen, and Guoqi Li. Improving graph collaborative filtering via spike signal embedding perturbation. *IEEE Transactions on Cognitive and Developmental Systems*, 2023.

[Ma *et al.*, 2024] De Ma, Xiaofei Jin, Shichun Sun, Yitao Li, Xundong Wu, Youneng Hu, Fangchao Yang, Huajin Tang, Xiaolei Zhu, Peng Lin, et al. Darwin3: a large-scale neuromorphic chip with a novel isa and on-chip learning. *National Science Review*, 11(5):nwae102, 2024.

[Maass, 1997] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[Rueckauer *et al.*, 2017] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[Shin *et al.*, 2024] Yehjin Shin, Jeongwhan Choi, Hyowon Wi, and Noseong Park. An attentive inductive bias for sequential recommendation beyond the self-attention. In

Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 8984–8992, 2024.

[Steck, 2019] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257, 2019.

[Sun *et al.*, 2022] Zhu Sun, Hui Fang, Jie Yang, Xinghua Qu, Hongyang Liu, Di Yu, Yew-Soon Ong, and Jie Zhang. Daisyrec 2.0: Benchmarking recommendation for rigorous evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[Tang and Wang, 2018] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[Venkataramani *et al.*, 2016] Swagath Venkataramani, Kaushik Roy, and Anand Raghunathan. Efficient embedded learning for iot devices. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 308–311. IEEE, 2016.

[Wang *et al.*, 2019] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: challenges, progress and prospects. *arXiv preprint arXiv:2001.04830*, 2019.

[Wang *et al.*, 2021] Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pages 1785–1797, 2021.

[Xia *et al.*, 2022] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Guandong Xu, and Quoc Viet Hung Nguyen. On-device next-item recommendation with self-supervised knowledge distillation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 546–555, 2022.

[Xia *et al.*, 2023] Xin Xia, Junliang Yu, Qinyong Wang, Chaoqun Yang, Nguyen Quoc Viet Hung, and Hongzhi Yin. Efficient on-device session-based recommendation. *ACM Transactions on Information Systems*, 41(4):1–24, 2023.

[Xia *et al.*, 2024] Tengxi Xia, Ju Ren, Wei Rao, Qin Zu, Wenjie Wang, Shuai Chen, and Yaoxue Zhang. Aerorec: an efficient on-device recommendation framework using federated self-supervised knowledge distillation. In *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*, pages 121–130. IEEE, 2024.

[Xing *et al.*, 2024] Xingrun Xing, Zheng Zhang, Ziyi Ni, Shitao Xiao, Yiming Ju, Siqi Fan, Yequan Wang, Jiajun Zhang, and Guoqi Li. Spikelm: Towards general spike-driven language modeling via elastic bi-spiking mechanisms. *arXiv preprint arXiv:2406.03287*, 2024.

[Yao *et al.*, 2022] Jiangchao Yao, Feng Wang, Xichen Ding, Shaohu Chen, Bo Han, Jingren Zhou, and Hongxia Yang. Device-cloud collaborative recommendation via meta controller. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4353–4362, 2022.

[Yao *et al.*, 2023] Man Yao, Guangshe Zhao, Hengyu Zhang, Yifan Hu, Lei Deng, Yonghong Tian, Bo Xu, and Guoqi Li. Attention spiking neural networks. *IEEE transactions on pattern analysis and machine intelligence*, 45(8):9393–9410, 2023.

[Yao *et al.*, 2024] Man Yao, Ole Richter, Guangshe Zhao, Ning Qiao, Yannan Xing, Dingheng Wang, Tianxiang Hu, Wei Fang, Tugba Demirci, Michele De Marchi, et al. Spike-based dynamic computing with asynchronous sensing-computing neuromorphic chip. *Nature Communications*, 15(1):4464, 2024.

[Yin *et al.*, 2023] Qing Yin, Hui Fang, Zhu Sun, and Yew-Soon Ong. Understanding diversity in session-based recommendation. *ACM Transactions on Information Systems*, 42(1):1–34, 2023.

[Yin *et al.*, 2024] Hongzhi Yin, Liang Qu, Tong Chen, Wei Yuan, Ruiqi Zheng, Jing Long, Xin Xia, Yuhui Shi, and Chengqi Zhang. On-device recommender systems: A comprehensive survey. *arXiv preprint arXiv:2401.11441*, 2024.

[Yuan *et al.*, 2020] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1469–1478, 2020.

[Yue *et al.*, 2024] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 930–938, 2024.

[Zhou *et al.*, 2022] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced mlp is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*, pages 2388–2399, 2022.

[Zhu *et al.*, 2022] Zulun Zhu, Jiaying Peng, Jintang Li, Liang Chen, Qi Yu, and Siqiang Luo. Spiking graph convolutional networks. In *International Joint Conference on Artificial Intelligence*, 2022.

[Zivic *et al.*, 2024] Pablo Zivic, Hernan Vazquez, and Jorge Sánchez. Scaling sequential recommendation models with transformers. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1567–1577, 2024.