MUFFIN: Mixture of User-Adaptive Frequency Filtering for Sequential Recommendation

Ilwoong Baek*
Sungkyunkwan University
Suwon, Republic of Korea
alltun100@skku.edu

Mincheol Yoon*
Sungkyunkwan University
Suwon, Republic of Korea
yoon56@skku.edu

Seongmin Park
Sungkyunkwan University
Suwon, Republic of Korea
psm1206@skku.edu

Jongwuk Lee†
Sungkyunkwan University
Suwon, Republic of Korea
jongwuklee@skku.edu

Abstract

Sequential recommendation (SR) aims to predict users' subsequent interactions by modeling their sequential behaviors. Recent studies have explored *frequency domain analysis*, which effectively models periodic patterns in user sequences. However, existing frequency-domain SR models still face two major drawbacks: (i) *limited frequency band coverage*, often missing critical behavioral patterns in a specific frequency range, and (ii) *lack of personalized frequency filtering*, as they apply an identical filter for all users regardless of their distinct frequency characteristics. To address these challenges, we propose a novel frequency-domain model, ***M**ixture of **U**ser-adaptive **F**requency **FI**lteri**N**g (**MUFFIN**)*, operating through two complementary modules. (i) The *global filtering module (GFM)* handles the entire frequency spectrum to capture comprehensive behavioral patterns. (ii) The *local filtering module (LFM)* selectively emphasizes important frequency bands without excluding information from other ranges. (iii) In both modules, the *user-adaptive filter (UAF)* is adopted to generate user-specific frequency filters tailored to individual unique characteristics. Finally, by aggregating both modules, MUFFIN captures diverse user behavioral patterns across the full frequency spectrum. Extensive experiments show that MUFFIN consistently outperforms state-of-the-art frequency-domain SR models over five benchmark datasets. The source code is available at https://github.com/ilwoong100/MUFFIN.

ACM Reference Format:
Ilwoong Baek, Mincheol Yoon, Seongmin Park, and Jongwuk Lee. 2025. MUFFIN: Mixture of User-Adaptive Frequency Filtering for Sequential Recommendation. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746252.3761190

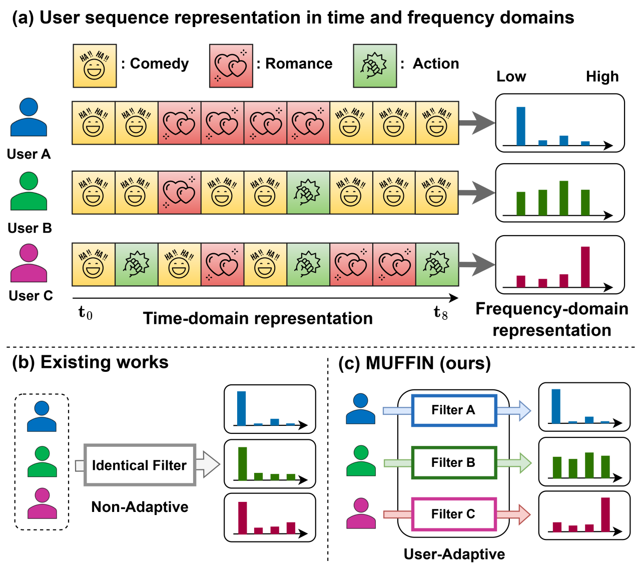


Figure 1: (a) Illustration of user sequences converted to frequency domain. Each user exhibits distinct frequency characteristics. (b) Existing works apply an identical filter to users, which fails to capture individual frequency characteristics. (c) Our model performs user-adaptive frequency filtering tailored to individual user characteristics.

1 Introduction

Sequential recommendation (SR) [3, 4, 37] aims to predict users' next interactions by modeling their historical behavior sequences. Unlike traditional recommendation settings that treat user-item interactions as independent instances, SR models [14, 23, 29, 32, 33, 41, 47] focus on capturing temporal dynamics and evolving user preferences in user sequences. These user sequences often exhibit intricate patterns at multiple levels of granularity, ranging from high-level transitions in user interests to subtle contextual variations [6, 36, 38]. Therefore, the core challenge of SR lies in effectively modeling diverse patterns in user interactions over time.

Recent advances in deep learning have significantly enhanced the SR performance, particularly by adopting transformer-based models using self-attention mechanisms. These models [14, 32] excel at modeling long-range dependencies within user sequences, effectively identifying item-to-item relationships. Despite their strengths, recent studies [22, 29] have highlighted a key limitation: self-attention mechanisms often struggle to capture fine-grained behavioral patterns, thereby impeding their ability to fully model the diverse and dynamic nature of user behavior.

To address this limitation, an emerging line of research [1, 2, 29, 40, 48] has explored *frequency-domain representation.* It is motivated

[*] Both authors contributed equally to this research.
[†] Corresponding author

by the observation that user sequences often exhibit inherent periodicities and multi-scale variations. Using the discrete Fourier transform (DFT), a time-domain representation is transformed into the frequency-domain representation. Therefore, complex behavioral patterns are decomposed into interpretable spectral components, where low-frequency signals correspond to gradual shifts in user interests, and high-frequency signals capture abrupt or short-term interest changes [27].

As illustrated in Figure 1(a), three users show distinct frequency characteristics based on their behavioral patterns. User A (blue) shows gradual transitions (*i.e.*, two changes) with dominant low-frequency components, and User B (green) shows a multi-band spectrum with both strong low- and high-frequency components, as comedy films dominate their consumption while intermittently exploring other genres (*i.e.*, four changes). Meanwhile, User C (pink) exhibits rapid genre switching with frequent item transitions (*i.e.*, seven changes), resulting in a frequency spectrum dominated by high-frequency components.

Although existing frequency-domain SR models [1, 29, 48] attempt to capture diverse user interaction patterns, they still suffer from two limitations. (i) *Limited frequency band coverage*: existing models face a trade-off between full frequency spectrum analysis and detailed frequency band-specific modeling. As observed in Figure 1(a), it is necessary to integrate global spectrum understanding and local band-specific analysis simultaneously. However, existing models cannot achieve this dual capability. (ii) *Lack of personalized frequency filtering*: existing models apply identical frequency filters regardless of the different frequency characteristics of users. As shown in Figure 1(b), when a low-frequency-emphasized filter is equally applied to three users, capturing each user's unique behavioral pattern is difficult, leading to information dilution.

To address this challenge, we propose *Mixture of **U**ser-adaptive **F**requency **FI**lteri**N**g (**MUFFIN**)*, which operates through two complementary modules. The *global filtering module (GFM)* processes the entire frequency spectrum simultaneously to capture comprehensive user characteristics across all frequency bands, capturing the users' overall behavioral tendencies. In contrast, the *local filtering module (LFM)* divides the spectrum into distinct frequency bands and selectively emphasizes the important ones while preserving information from the others. To enhance user adaptiveness for each module, we also introduce a *user-adaptive filter (UAF)* that dynamically generates personalized filters tailored to each user's unique characteristics. Finally, MUFFIN ensures personalized filtering that adapts to diverse user behavior patterns, enabling both modules to adjust their filtering strength based on individual user patterns. As depicted in Figure 1(c), MUFFIN generates user-adaptive filters and effectively identifies individual behavioral patterns. Through extensive experiments, MUFFIN is compared against eight SR models and outperforms state-of-the-art frequency-domain models over five benchmark datasets.

The key contributions of this paper are summarized as follows:

- We propose a novel dual filtering architecture consisting of the *global filtering module (GFM)* and the *local filtering module (LFM)*. The GFM processes the entire frequency spectrum to capture comprehensive behavioral patterns across all frequency ranges, while the LFM selectively emphasizes specific frequency bands to model user-specific behavioral patterns without losing any frequency information.
- We employ the *user-adaptive filter (UAF)* that serves as a core mechanism. It enables both GFM and LFM to perform user-specific filtering by generating personalized filters based on individual frequency-domain representations.
- Through extensive experiments on five datasets over eight SR models, we demonstrate that MUFFIN achieves over state-of-the-art models, showcasing its ability to adapt to user-specific patterns and capture diverse behavior preferences.

2 Preliminaries

2.1 Problem Formulation

Let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items, respectively, where $|\mathcal{U}|$ and $|\mathcal{I}|$ are the total number of users and items. For each user $u \in \mathcal{U}$, we are given a user sequence $S_u = [i_1, i_2, ..., i_{|S_u|}]$, where $i_j \in \mathcal{I}$ indicates the j-th item in the user sequence, and $|S_u|$ is the length of the user sequence. Given a user sequence S_u, the goal of SR is to predict the next item $i_{|S_u|+1}$ with which user u will most likely interact. It is formulated as training an optimal model parameter θ^* that maximizes the conditional probability:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, P_\theta(i = i_{|S_u|+1} \mid S_u). \tag{1}$$

For each prediction, the SR model parameterized by θ produces a probability distribution over the item set $\mathcal{I}$. Thus, higher probabilities indicate items the user is more likely to interact with next.

2.2 Fourier Transform

The discrete Fourier transform (DFT) is a fundamental component of digital signal processing [25, 30], converting a sequence in the time domain into the frequency domain. Given an input sequence $\mathbf{S} \in \mathbb{R}^N$ with length N, the DFT is denoted as $\mathcal{F} : \mathbb{R}^N \to \mathbb{C}^N$, and its inverse, *i.e.*, the inverse discrete Fourier transform (IDFT), is denoted as $\mathcal{F}^{-1} : \mathbb{C}^N \to \mathbb{R}^N$. The DFT can be performed as:

$$\mathbf{F} = \mathcal{F}(\mathbf{S}) = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{-2\pi i}{N}} & \cdots & e^{\frac{-2\pi i(N-1)}{N}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & e^{\frac{-2\pi i(N-1)}{N}} & \cdots & e^{\frac{-2\pi i(N-1)^2}{N}} \end{bmatrix} \mathbf{S}, \tag{2}$$

where i is the imaginary unit, and $\mathbf{F} \in \mathbb{C}^N$ is the frequency component of the sequence $\mathbf{S}$. In practice, we utilize the Fast Fourier Transform (FFT) [5] algorithm, which efficiently computes the DFT with a computational complexity of $O(N \log N)$ compared to the direct DFT computation of $O(N^2)$. Furthermore, since our sequence $\mathbf{S}$ consists of real numbers, its Fourier transform exhibits conjugate symmetry:

$$\mathbf{F}[k] = \mathbf{F}[N-k]^*, k = 1, 2, ..., \lfloor N/2 \rfloor, \tag{3}$$

where * denotes the complex conjugate. Due to this symmetry property, we use the real-valued fast Fourier transform (RFFT) [21, 31], which computes only the non-redundant half of the spectrum. This reduces the output dimension from N to $\lfloor \frac{N}{2} \rfloor$+1 while preserving all unique frequency information. Using RFFT, the frequency-domain representation is:

$$\mathbf{F} = \mathcal{F}_R(\mathbf{S}) \in \mathbb{C}^M, \text{ where } M = \left\lfloor \frac{N}{2} \right\rfloor + 1, \tag{4}$$

where $\mathcal{F}_R(\cdot)$ denotes RFFT. This reduces computational overhead and maintains all essential frequency components needed for SR. For notational convenience, we denote the RFFT $\mathcal{F}_R$ as $\mathcal{F}$.

To capture the frequency band in $\mathbf{F}$, we can extract the frequency components from indices f_t to f_{t+1}, where f_t denotes the starting index of the t-th frequency band. It represents the particular frequency band in the transformed sequence.

$$\mathbf{B}_t = \mathbf{F}[f_t : f_{t+1}], \tag{5}$$

where $\mathbf{B}_t \in \mathbb{C}^{f_{t+1}-f_t}$ and $[:]$ indicates the slice operation. In $\mathbf{B}_t$, smaller values of t correspond to low frequencies, while larger values of t correspond to high frequencies.

2.3 Learnable Frequency Filter

The filtering layer in the frequency domain is a key component that adjusts the importance of frequency components for input sequences. This is based on the concept of frequency filtering widely used in signal processing, and some studies [1, 48] attempted to implement it using learnable filters in SR. The learnable frequency filter layer is defined as follows:

$$f(\mathbf{S}) = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{S}) \odot \mathbf{W}), \tag{6}$$

where $\mathcal{F}^{-1}$ represents inverse RFFT, $\mathbf{W} \in \mathbb{C}^M$ is learnable frequency filter, and $\odot$ denotes element-wise multiplication.

This filter $\mathbf{W}$ is optimized during model training. In other words, it captures specific patterns in the sequence by emphasizing or suppressing certain frequency bands.

3 Proposed Model: MUFFIN

In this section, we present ***M**ixture of **U**ser-adaptive **F**requency **FI**lteri**N**g (MUFFIN)*, capturing diverse user behavior patterns. As depicted in Figure 2, it comprises two complementary filtering modules: *global filtering module (GFM)* and *local filtering module (LFM)*, each designed to extract different aspects of user behavior. To enable user adaptiveness in both modules, we also incorporate a *user-adaptive filter (UAF)*, which dynamically generates personalized filters tailored to each user's frequency-domain characteristics. Section 3.1 details these key components of MUFFIN, and Section 3.2 describes the training procedure of MUFFIN.

3.1 Mixture of Filtering Modules

Embedding layer. The item embedding matrix $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}|\times d}$, where d is the embedding dimension, is employed to project items into a latent embedding space. Given a user sequence $S_u = [i_1, i_2, \ldots, i_{|S_u|}]$, each item i_j is represented as its corresponding embedding vector $\mathbf{e}_j \in \mathbb{R}^d$. These item embeddings are concatenated to form the initial sequence representations $\mathbf{H}^0 = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|S_u|}]$, where $\mathbf{H}^0 \in \mathbb{R}^{n\times d}$, and n is the maximum length of the sequence.

User-adaptive filter (UAF). It is dynamically generated to enable personalized filtering within global and local filtering modules. The initial sequence representation $\mathbf{H}^0$ is transformed from the time domain into the frequency domain by the Fourier transform:

$$\mathbf{X}^0 = \mathcal{F}(\mathbf{H}^0) \in \mathbb{C}^{m\times d}, \tag{7}$$

where $m = \lfloor n/2 \rfloor + 1$, and $\mathbf{X}^0$ denotes the initial representation in the frequency domain.

Inspired by recent advances in frequency-based filtering for computer vision [19], we design a convolution filter $h(\cdot)$ to capture user-specific behavior patterns:

$$\mathrm{C}(\mathbf{X}^0) = \text{BatchNorm1D}\left(\texttt{Conv1D}(\|\mathbf{X}^0\|^\top)\right) \in \mathbb{R}^{d\times m}, \tag{8}$$

$$h(\mathbf{X}^0) = \sigma\left(\mathrm{C}(\mathbf{X}^0)\right)^\top \in \mathbb{R}^{m\times d}, \tag{9}$$

where σ denotes sigmoid function, and $\|\cdot\|$ denotes the amplitude (*i.e.*, magnitude) of the complex-valued tensor. $\top$ indicates the matrix transpose operator. The `Conv1D` [17] operation employs a convolutional kernel $\mathbf{W}_c \in \mathbb{R}^{d\times d\times c}$, where c is a hyperparameter for kernel size. By leveraging amplitude information and applying 1D convolution, we compute a weighted aggregation of each frequency component and its neighboring frequencies. Therefore, this convolutional filter captures meaningful patterns in the frequency-amplitude domain to construct personalized representations.

Global filtering module (GFM). It captures high-level behavioral patterns by analyzing the full frequency spectrum of user sequences. At the l-th layer, the frequency-domain representation $\mathbf{X}^l_{global}$ is computed as:

$$\mathbf{X}^l_{global} = \mathcal{F}(\mathbf{H}^l_{global}), \tag{10}$$

where $\mathbf{H}^0_{global}$ corresponds to the initial representation $\mathbf{H}^0$.

To modulate frequency components across the entire spectrum, we adopt a learnable filter $\mathbf{W}_{global} \in \mathbb{C}^{m\times d}$. It is refined using the user-adaptive filter (UAF) as follows:

$$\tilde{\mathbf{W}}^l_{global} = h(\mathbf{X}^0) \odot \mathbf{W}^l_{global}, \tag{11}$$

where $h(\cdot)$ denotes the amplitude-based convolutional filter described in Eq. (9). Notably, the UAF takes only the initial frequency representation $\mathbf{X}^0$ as input, rather than the layer-specific input $\mathbf{X}^l$ for $l > 0$. This enables the filter to encode user-specific characteristics from the original sequence spectrum.

The global adaptive filter $\tilde{\mathbf{W}}^l_{global}$ is then applied to scale the frequency-domain representation. It is transformed back to the time domain via the inverse Fourier transform:

$$\tilde{\mathbf{H}}^l_{global} = \mathcal{F}^{-1}(\mathbf{X}^l_{global} \odot \tilde{\mathbf{W}}^l_{global}), \tag{12}$$

where $\tilde{\mathbf{H}}^l_{global} \in \mathbb{R}^{n\times d}$. To ensure stable training and effective gradient propagation [9], the final output of the GFM incorporates residual connections, dropout, and layer normalization:

$$\mathbf{O}^l_{global} = \text{LayerNorm}\left(\mathbf{H}^l_{global} + \text{Dropout}(\tilde{\mathbf{H}}^l_{global})\right). \tag{13}$$

Local filtering module (LFM). It is designed to capture fine-grained patterns by partitioning the frequency spectrum and emphasizing specific frequency bands. At the l-th layer, the frequency-domain representation of the user sequence $\mathbf{X}^l_{local}$ is obtained by applying the Fourier transform to the input sequence $\mathbf{H}^l_{local}$, and dividing the result into K contiguous frequency bands:

$$\mathbf{X}^l_{local} = [\mathbf{B}^l_1; \mathbf{B}^l_2; \ldots; \mathbf{B}^l_K] = \mathcal{F}(\mathbf{H}^l_{local}), \tag{14}$$

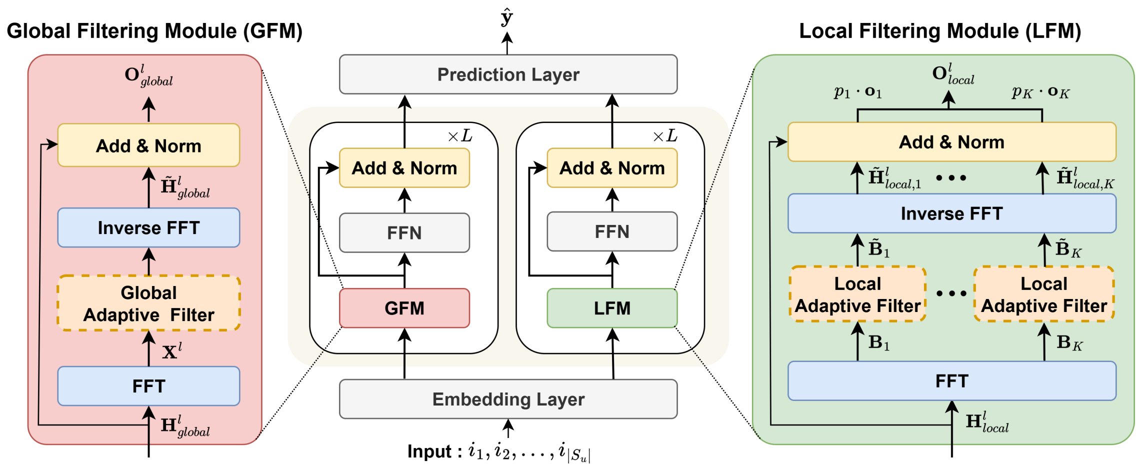


Figure 2: Overview of MUFFIN, which trains with two parallel modules: global filtering module (GFM) and local filtering module (LFM). Both modules utilize the user-adaptive filter (UAF) to adjust to individual user behavior patterns in the frequency domain. A mixture of the module outputs is used to predict the target item.

where each $\mathbf{B}^l_t = \mathbf{X}^l_{local}[f_t : f_{t+1}] \in \mathbb{C}^{a_t \times d}$ denotes the t-th frequency band with starting index $f_1 = 0$, and K is a hyperparameter to control the number of frequency bands. The band size a_t determines the number of frequency components assigned to each band [1]. This band-wise decomposition enables our model to focus on local frequency patterns, facilitating the extraction of subtle behavioral signals that may not be captured by global representations.

Similar to GFM, LFM also utilizes a learnable filter $\mathbf{W}_{local} \in \mathbb{C}^{m \times d}$ to control the contribution of each frequency component. To incorporate user-specific behavior, we apply UAF to generate a local adaptive filter at the l-th layer:

$$\tilde{\mathbf{W}}^l_{local} = h(\mathbf{X}^0) \odot \mathbf{W}^l_{local}. \tag{15}$$

This filter is then divided into K bands to match the band-wise structure of the frequency representation:

$$\tilde{\mathbf{W}}^l_{local} = [\tilde{\mathbf{W}}^l_{local,1}; \tilde{\mathbf{W}}^l_{local,2}; \ldots; \tilde{\mathbf{W}}^l_{local,K}], \tag{16}$$

where $\tilde{\mathbf{W}}^l_{local,t} \in \mathbb{C}^{a_t \times d}$ is applied to the t-th frequency band.

To preserve the full frequency resolution required for the inverse Fourier transform, we apply zero-padding after filtering each frequency band:

$$\begin{aligned}\tilde{\mathbf{B}}^l_t &= \text{ZeroPadding}(\mathbf{B}^l_t \odot \tilde{\mathbf{W}}^l_{local,t}) \\ &= [\mathbf{0}_{f_t \times d}; \mathbf{B}^l_t \odot \tilde{\mathbf{W}}^l_{local,t}; \mathbf{0}_{(m-f_{t+1}) \times d}] \in \mathbb{C}^{m \times d},\end{aligned} \tag{17}$$

where $\mathbf{0}_{f_t \times d}$ and $\mathbf{0}_{(m-f_{t+1}) \times d}$ are zero matrices that pad the beginning and end of the spectrum, respectively.

Each zero-padded frequency representation is transformed back to the time domain:

$$\tilde{\mathbf{H}}^l_{local,t} = \mathcal{F}^{-1}(\tilde{\mathbf{B}}^l_t) \in \mathbb{R}^{n \times d}. \tag{18}$$

[1]Here, $a_t = \lfloor \frac{t \times m}{K} \rfloor - \lfloor \frac{(t-1) \times m}{K} \rfloor$ and $\sum_{t=1}^{K} a_t = m$, ensuring the complete coverage of the frequency spectrum.

We then apply dropout, residual connection, and layer normalization to obtain the t-th frequency band output:

$$\mathbf{o}_t = \text{LayerNorm}\left(\mathbf{H}^l_{local} + \text{Dropout}(\tilde{\mathbf{H}}^l_{local,t})\right). \tag{19}$$

To integrate the outputs from all bands, we adopt a soft gating mechanism [12] that dynamically weighs each band's contribution based on the input representation in the frequency domain. Specifically, a gating function $g(\cdot)$ is used to produce a softmax distribution over all frequency bands:

$$p_t = \text{Softmax}(g(\|\mathbf{X}^l_{local}\|))_t, \tag{20}$$

where $\|\mathbf{X}^l_{local}\| \in \mathbb{R}^{m \times d}$ denotes the amplitude of the complex spectrum, and $p_t \in \mathbb{R}$ is the importance weight assigned to the t-th frequency band. For $g(\cdot)$, we use a three-layer MLP with GeLU activation function.

Unlike sparse gating methods such as top-k selection [28], we utilize a soft selection strategy to preserve the contributions of all bands. This avoids potential information loss from discarding seemingly less important frequencies that may still carry valuable user-specific signals.

Finally, the output of LFM is constructed through a weighted aggregation of frequency band outputs based on their learned significance:

$$\mathbf{O}^l_{local} = \sum_{t=1}^{K} p_t \cdot \mathbf{o}_t. \tag{21}$$

Feed forward network (FFN). It incorporates non-linearity into the model, enabling it to capture intricate patterns in user sequences.

$$\text{FFN}(\mathbf{X}) = \text{GeLU}(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{22}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d \times 4d}$, $\mathbf{W}_2 \in \mathbb{R}^{4d \times d}$ are weight matrices, and $\mathbf{b}_1 \in \mathbb{R}^{4d}$, $\mathbf{b}_2 \in \mathbb{R}^{d}$ are bias vectors.

We then apply the FFN to the outputs of both the global and local modules:

$$\mathbf{H}_*^{l+1} = \text{LayerNorm}\left(\mathbf{O}_*^l + \text{Dropout}\left(\text{FFN}(\mathbf{O}_*^l)\right)\right), \tag{23}$$

where $* \in \{global,\ local\}$.

The updated representations, $\mathbf{H}_{global}^{l+1}$ and $\mathbf{H}_{local}^{l+1}$, are passed into the next layer's GFM and LFM, respectively, allowing the model to progressively refine the sequence representations across layers. After passing the final layer L, we obtain the final outputs $\mathbf{H}_{global}^{L}$ and $\mathbf{H}_{local}^{L}$, respectively.

Prediction layer. This layer fuses the outputs of both global and local branches to form the final sequence representation used for next-item prediction. We concatenate the final global and local representations and project them through a linear transformation:

$$\mathbf{H}^L = [\mathbf{H}_{global}^L; \mathbf{H}_{local}^L]\mathbf{W}_p, \tag{24}$$

where $\mathbf{W}_p \in \mathbb{R}^{2d\times d}$ and $\mathbf{H}^L \in \mathbb{R}^{n\times d}$.

To preserve information from the original sequence embeddings, we add a residual connection with the initial embedding $\mathbf{H}^0$, followed by dropout and layer normalization:

$$\hat{\mathbf{H}}^L = \text{Dropout}\left(\text{LayerNorm}(\mathbf{H}^0 + \mathbf{H}^L)\right). \tag{25}$$

Finally, we select the representation of the last interacted item $\hat{\mathbf{H}}_{|S^u|}^L \in \mathbb{R}^d$ and compute the predicted scores for all items.

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{E}\hat{\mathbf{H}}_{|S^u|}^L) \in \mathbb{R}^{|\mathcal{I}|}, \tag{26}$$

where $\mathbf{E} \in \mathbb{R}^{|\mathcal{I}|\times d}$ is the whole item embedding matrix shared with the input embedding matrix.

3.2 Model Training

We train MUFFIN using a multi-objective loss that encourages accurate next-item prediction while promoting the complementary strengths of its dual-module architecture.

Recommendation loss. The main training objective is to minimize the cross-entropy loss between the predicted distribution $\hat{\mathbf{y}}$ and the ground truth one-hot vector $\mathbf{y}$:

$$\mathcal{L}_{rec} = -\sum_{i=1}^{|\mathcal{I}|} \mathbf{y}_i \log(\hat{\mathbf{y}}_i). \tag{27}$$

Auxiliary training loss. Although the dual-branch design enables MUFFIN to capture both global and local frequency signals, relying solely on the final fused output may not fully exploit the specialization of each module. Motivated by prior work [35], we introduce auxiliary supervision that independently encourages each module to perform next-item prediction.

We compute predictions from the final layer representations of each module:

$$\hat{\mathbf{y}}^{local} = \text{Softmax}(\mathbf{E}\mathbf{H}_{local}^L), \tag{28}$$

$$\hat{\mathbf{y}}^{global} = \text{Softmax}(\mathbf{E}\mathbf{H}_{global}^L). \tag{29}$$

The auxiliary loss is defined as the sum of their predictions. This loss encourages global and local modules to develop distinct yet predictive representations.

$$\mathcal{L}_{aux} = -\sum_{i=1}^{|\mathcal{I}|} \left(\mathbf{y}_i \log(\hat{\mathbf{y}}_i^{local}) + \mathbf{y}_i \log(\hat{\mathbf{y}}_i^{global})\right). \tag{30}$$

Load balancing loss. Inspired by a study [44], we introduce a load-balancing regularization loss to prevent the soft gate in LFM from focusing disproportionately on a small subset of frequency bands. This loss encourages uniform attention across all K frequency bands by penalizing deviation from the uniform distribution:

$$\mathcal{L}_{bal} = \frac{1}{K}\sum_{t=1}^{K} \|p_t - \frac{1}{K}\|_2^2, \tag{31}$$

where p_t is the soft gate probability of the t-th band in Eq. (20). This promotes diversity and full utilization of the frequency spectrum during training.

Total loss. The overall training objective of MUFFIN combines three loss functions:

$$\mathcal{L} = \mathcal{L}_{rec} + \alpha\mathcal{L}_{aux} + \beta\mathcal{L}_{bal}, \tag{32}$$

where α and β are hyperparameters that control the strength of the auxiliary and load balancing losses, respectively.

Table 1: Data statistics after preprocessing. 'Avg. Length' indicates the average number of interactions per user.

Dataset	Beauty	Toys	Sports	Yelp	ML-1M
# Users	22,363	19,412	35,598	30,499	6,041
# Items	12,101	11,924	18,357	20,068	3,417
# Inter.	198,502	167,597	296,337	317,182	999,611
Avg. Length	8.9	8.6	8.3	10.4	165.5
Sparsity	99.93%	99.93%	99.95%	99.95%	95.16%

4 Experiment Setup

Datasets. We evaluate MUFFIN on five widely used benchmark datasets for sequential recommendation (SR), following the standard setup in prior work [1]. These datasets span diverse domains – e-commerce, local services, and media – providing a comprehensive validation of the model's effectiveness. **Amazon Beauty, Toys, and Sports**[2] [20] are subsets of the Amazon product review corpus. Each dataset contains timestamped user-item interactions derived from product reviews and ratings. **Yelp**[3] consists of user interactions with local businesses, including reviews and ratings. **ML-1M**[4] contains 1 million movie ratings from 6,000 users on 4,000 movies, along with user demographics and movie metadata information.

Each dataset is preprocessed following established protocols to ensure fair and consistent comparisons with prior work [1, 2, 32]. Specifically, we use *five-core settings* by removing users and items that occur less than five times in the dataset. Table 1 reports the detailed statistics of datasets.

Competing models. We compare MUFFIN against eight SR baselines, including time-domain and frequency-domain SR models.

[2]https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html
[3]https://www.yelp.com/dataset
[4]https://grouplens.org/datasets/movielens/

Table 2: Accuracy comparison for eight SR models on five datasets. The best and second-best results are marked in bold and $\underline{\text{underlined}}$. † is $p < 0.05$ in a one-tailed t-test, and 'Imp.' is improvement ratio, both compared with the second-best model.

Dataset	Metric	GRU4Rec	SASRec	BERT4Rec	DuoRec	FMLPRec	FEARec	BSARec	SLIME4Rec	**MUFFIN**	Imp. (%)
Beauty	R@5	0.0399	0.0563	0.0338	0.0570	0.0544	$\underline{0.0580}$	0.0557	0.0579	**0.0592**†	**2.19**
	R@10	0.0609	0.0843	0.0484	0.0863	0.0861	0.0863	0.0874	$\underline{0.0885}$	**0.0919**†	**3.80**
	R@20	0.0886	0.1186	0.0696	0.1216	0.1247	0.1227	0.1256	$\underline{0.1273}$	**0.1304**†	**2.44**
	N@5	0.0266	0.0329	0.0232	0.0352	0.0321	0.0360	0.0334	$\underline{0.0361}$	**0.0381**†	**5.45**
	N@10	0.0333	0.0418	0.0280	0.0446	0.0424	0.0451	0.0437	$\underline{0.0460}$	**0.0487**†	**5.80**
	N@20	0.0403	0.0505	0.0333	0.0535	0.0521	0.0545	0.0533	$\underline{0.0558}$	**0.0583**†	**4.60**
Toys	R@5	0.0339	0.0625	0.0327	0.0653	0.0597	0.0670	0.0636	$\underline{0.0670}$	**0.0692**†	**3.28**
	R@10	0.0510	0.0898	0.0451	0.0946	0.0901	0.0965	0.0944	$\underline{0.0995}$	**0.1005**	**0.97**
	R@20	0.0740	0.1226	0.0620	0.1296	0.1269	0.1307	0.1310	$\underline{0.1365}$	**0.1382**	**1.22**
	N@5	0.0235	0.0352	0.0231	0.0388	0.0343	0.0395	0.0379	$\underline{0.0401}$	**0.0441**†	**10.06**
	N@10	0.0290	0.0440	0.0271	0.0483	0.0441	0.0490	0.0478	$\underline{0.0506}$	**0.0542**†	**7.25**
	N@20	0.0347	0.0522	0.0313	0.0571	0.0534	0.0576	0.0571	$\underline{0.0599}$	**0.0641**†	**7.02**
Sports	R@5	0.0225	0.0303	0.0118	0.0322	0.0322	0.0286	0.0320	$\underline{0.0341}$	**0.0359**†	**5.18**
	R@10	0.0362	0.0476	0.0206	0.0498	0.0511	0.0434	0.0511	$\underline{0.0530}$	**0.0570**†	**7.48**
	R@20	0.0566	0.0690	0.0345	0.0734	0.0759	0.0635	0.0751	$\underline{0.0788}$	**0.0841**†	**6.64**
	N@5	0.0150	0.0167	0.0073	0.0198	0.0178	0.0176	0.0177	$\underline{0.0210}$	**0.0213**	**1.43**
	N@10	0.0193	0.0223	0.0101	0.0254	0.0239	0.0223	0.0239	$\underline{0.0271}$	**0.0281**†	**3.82**
	N@20	0.0244	0.0276	0.0136	0.0313	0.0301	0.0274	0.0299	$\underline{0.0336}$	**0.0349**†	**3.87**
Yelp	R@5	0.0253	0.0425	0.0192	0.0435	0.0471	0.0399	0.0458	$\underline{0.0473}$	**0.0510**†	**7.75**
	R@10	0.0415	0.0597	0.0341	0.0634	0.0681	0.0567	0.0691	$\underline{0.0729}$	**0.0755**	**3.66**
	R@20	0.0671	0.0862	0.0570	0.0929	0.0998	0.0817	0.1030	$\underline{0.1097}$	**0.1111**	**1.31**
	N@5	0.0166	0.0322	0.0120	0.0317	$\underline{0.0344}$	0.0287	0.0325	0.0323	**0.0350**†	**1.65**
	N@10	0.0217	0.0377	0.0167	0.0381	$\underline{0.0412}$	0.0341	0.0400	0.0405	**0.0429**†	**4.21**
	N@20	0.0282	0.0444	0.0225	0.0455	0.0491	0.0404	0.0485	$\underline{0.0498}$	**0.0518**†	**3.83**
ML-1M	R@5	0.2085	$\underline{0.2182}$	0.1846	0.2168	0.2160	0.1843	0.2168	0.2158	**0.2252**†	**3.22**
	R@10	0.2925	$\underline{0.3103}$	0.2722	0.3098	0.3049	0.2709	0.3090	0.3044	**0.3187**†	**2.71**
	R@20	0.3924	$\underline{0.4121}$	0.3823	0.4099	0.4078	0.3829	0.4117	0.4080	**0.4179**†	**1.42**
	N@5	0.1436	0.1495	0.1222	$\underline{0.1497}$	0.1487	0.1237	0.1485	0.1479	**0.1546**†	**3.27**
	N@10	0.1707	0.1791	0.1505	$\underline{0.1798}$	0.1774	0.1517	0.1782	0.1765	**0.1848**†	**2.78**
	N@20	0.1959	0.2049	0.1783	$\underline{0.2050}$	0.2034	0.1799	0.2042	0.2026	**0.2098**†	**2.36**

Time-domain SR models: **GRU4Rec** [11] utilizes a gated recurrent unit to model users' sequential behaviors. **SASRec** [14] leverages the self-attention mechanism to model item-item dependencies within sequences. **BERT4Rec** [32] applies BERT's bidirectional encoding structure to SR, improving the contextual understanding of sequences through masked item prediction. **DuoRec** [24] enhances sequence representations by incorporating contrastive self-supervised learning tasks into the transformer architecture.

Frequency-domain SR models: **FMLP-Rec** [48] is a pioneer frequency-domain SR model using element-wise complex weight filters to process frequency components. **FEARec** [2] integrates frequency-domain signals directly into attention computation, enabling the model to leverage time and frequency information. **BSARec** [29] adjusts the influence on the high-frequency domains and combines it with an inductive bias of the self-attention mechanism. **SLIME4Rec** [1] utilizes a frequency ramp structure with layer-wise dynamic and static frequency band selection to capture diverse sequential patterns.

Evaluation protocol. We evaluate the performance of the next-item prediction task using two standard metrics: Recall@K and NDCG@K, where K is set to {5, 10, 20}. For brevity, we denote these metrics as R@K and N@K, respectively. Following prior works [14, 26], we adopt the *leave-one-out evaluation* strategy. For each user's interaction sequence, the last item is held out as the test instance, the second-to-last item is used for validation, and the remaining interactions form the training set. We rank the ground-truth item against all other items (including those that appeared in the training set) and compute the ranking-based metrics on the test set. The final results are the average scores across all test users. All the results are averaged over five runs with different random seeds.

Implementation details. All experiments were conducted using the Recbole[5] [46] and Recbole-DA[6] frameworks, open-source libraries for building and evaluating SR models. MUFFIN was implemented on Recbole-DA to ensure a fair comparison, as it is the same experimental environment used in SLIME4Rec [1]. For all models, we adopt the Adam optimizer [16] with a learning rate of 0.001, a batch size of 256, and a hidden dimension of 64. To ensure consistency and comparability with prior works [1, 2, 14, 29, 48], all frequency-based models are set to use two layers. To train MUFFIN, we conduct a grid search for hyperparameters: the number of frequency bands $K \in \{2, 4, 6, 8, 10\}$, the auxiliary loss weight $\alpha \in \{0.05, 0.1, 0.2, 0.5, 1\}$, and load balancing loss weight $\beta \in \{0.05, 0.1, 0.2, 0.5, 1\}$. The dropout rate is set to 0.1 for ML-1M and 0.4 for the remaining datasets. We searched for the kernel size

[5]https://github.com/RUCAIBox/RecBole
[6]https://github.com/RUCAIBox/RecBole-DA

c for the UAF module over {3, 5, 7}. All baseline models were trained with the hyperparameters reported in their original papers to ensure optimal performance. We adopt the cross-entropy loss function across all models, as it consistently outperforms BCE and BPR loss in prior studies. Following standard practice [2, 29], the maximum sequence length is set to 50. N@20 is used as the validation metric, and early stopping is applied if performance does not improve for 15 consecutive epochs. All experiments were conducted on a server equipped with an NVIDIA RTX 3090 24GB GPU and an Intel Xeon Gold 6226R CPU.

5 Experimental Results

5.1 Overall Performance

Table 2 reports the performance of MUFFIN and other baseline models across five datasets. All improvement ratios are computed from exact values without rounding. Our key findings are as follows.

Outstanding performance of MUFFIN. MUFFIN consistently achieves state-of-the-art performance across all datasets. Compared to DuoRec [24] as the strongest transformer-based baseline, MUFFIN achieves average relative improvements of 9.84% in R@10 and 9.48% in N@10. Against SLIME4Rec [1] as the strongest frequency-domain competitor, MUFFIN achieves average gains of 4.12% in R@10 and 5.48% in N@10. These improvements underscore MUFFIN's effectiveness in overcoming the limitations of prior models via its dual filtering architecture, simultaneously capturing both global and local behavioral patterns.

Time-domain vs. Frequency-domain models. Traditional time-domain models like, SASRec [14] and DuoRec [24] achieve competitive results. Their performances are particularly pronounced on ML-1M with long user sequences. In this situation, their superior capability to model long-range dependencies proves advantageous. Recent frequency-domain models show promising advancements over traditional time-domain models. Among them, SLIME4Rec [1] stands out, outperforming most time-domain models on four out of five datasets. This highlights the potential of frequency-aware modeling in capturing intricate user behavior patterns.

5.2 Ablation Study

To evaluate the effect of each component in MUFFIN, we conduct extensive ablation studies on the Beauty and ML-1M datasets. Table 3 reports the comparative results of the original model and its ablated variants.

Effect of key model components. Removing any core component in MUFFIN resulted in consistent performance degradation, highlighting the importance of each architectural element. Notably, substituting user-adaptive filtering (UAF) with fixed non-adaptive filters (*w/o UAF*) leads to performance drops, underscoring the value of personalized frequency selection. Additionally, eliminating either the Global Filtering Module (GFM) or the Local Filtering Module (LFM) (*w/o GFM* or *w/o LFM*) significantly reduces performance compared to the complete dual-module configuration. Based on these results, we confirm that global and local frequency perspectives are complementary and jointly necessary for modeling complex user behavior.

Table 3: Ablation study of MUFFIN on Beauty and ML-1M. 'w/o' denotes the model variant without the corresponding component.

Model	Beauty		ML-1M	
	R@10	N@10	R@10	N@10
MUFFIN	0.0919	0.0487	0.3187	0.1848
w/o UAF	0.0906	0.0473	0.3151	0.1803
w/o GFM	0.0887	0.0460	0.3086	0.1759
w/o LFM	0.0906	0.0467	0.3079	0.1798
w/o $\mathcal{L}_{aux}$	0.0862	0.0471	0.3101	0.1813
w/o $\mathcal{L}_{bal}$	0.0907	0.0475	0.3151	0.1810

Table 4: Accuracy comparison between MUFFIN and MUFFIN$_{\text{MLP}}$ on Beauty and ML-1M. MUFFIN$_{\text{MLP}}$ replaces the convolution layers with MLP layers in UAF.

Model	Beauty		ML-1M	
	R@10	N@10	R@10	N@10
MUFFIN (ours)	0.0919	0.0487	0.3187	0.1848
MUFFIN$_{\text{MLP}}$	0.0913	0.0476	0.3115	0.1810

Effect of loss functions. The auxiliary training loss is also critical to the model's effectiveness. Removing the auxiliary loss $\mathcal{L}aux$, which provides independent supervision to GFM and LFM, results in the most substantial drop in performance. This highlights its importance in promoting the distinct learning objectives of the two modules. Similarly, removing the load balancing loss $\mathcal{L}bal$ degrades performance, indicating its role in ensuring effective and balanced utilization of frequency bands. Further analysis of the load balancing mechanism is provided in Section 5.3.

Effect of convolutional filters in UAF. To validate the effectiveness of the convolutional filter design in UAF, we conduct an additional study by replacing the convolution layers with MLP layers (denoted as MUFFIN$_{\text{MLP}}$). As reported in Table 4, MUFFIN using convolutional filters consistently outperforms MUFFIN$_{\text{MLP}}$ on both datasets. Unlike MLPs that process each frequency component independently, convolutional filters better capture local frequencies and spatial patterns in the frequency spectrum. This capability makes MUFFIN more effective for generating user-specific filters.

5.3 In-depth Analysis

Visualization of different frequency filters. To investigate how user-adaptive frequency filtering in MUFFIN differs from existing models, we analyze the learned frequency filters of representative models. Figure 3 shows the user-adaptive filters of GFM and LFM in MUFFIN compared to the frequency filters of FMLPRec [48] and SLIME4Rec [1]. We found two interesting findings as follows. (i) FMLPRec [48] exhibits different tendencies across datasets. It emphasizes low-frequency components (indices 0–5) in Beauty while showing a gradually increasing pattern toward high-frequency regions in ML-1M. SLIME4Rec [1] completely blocks certain frequency bands (*i.e.*, zero amplitude) in Beauty while maintaining intermediate values across all frequency bands in ML-1M. Moreover, both FMLPRec [48] and SLIME4Rec [1] employ identical filters for

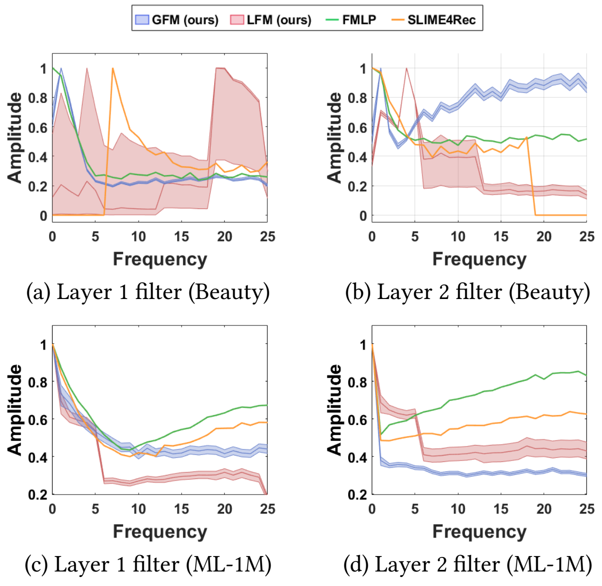


Figure 3: Frequency filter amplitude distributions of MUFFIN's modules and baseline models on Beauty and ML-1M. The x-axis represents frequency indices, and the y-axis represents filter amplitude values. Shaded areas for MUFFIN's modules illustrate the variation in amplitude across different users for each frequency component.

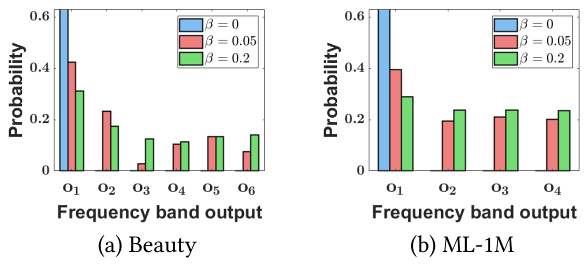


Figure 4: Probability distribution of frequency band outputs in LFM across different load balancing weights β on Beauty and ML-1M.

all users. (ii) In clear contrast, both GFM and LFM in MUFFIN exhibit variability in filter weights even at the same frequency components, indicating dynamically generated user-adaptive filters. Specifically, GFM maintains relatively stable patterns over the full frequency spectrum while incorporating user-specific variations. Meanwhile, on Beauty, LFM assigns highly diverse weights ranging from near zero to one across low and high-frequency bands among users. On ML-1M, LFM shows user-specific variations predominantly in mid to high-frequency bands. These findings highlight that MUFFIN mitigates the static filtering of existing frequency-domain models by dynamically generating user-adaptive frequency filters that reflect each user's unique frequency characteristics.

Local frequency band probability. We analyze the probability distributions of frequency bands in LFM under varying load-balancing weights β. Figure 4 depicts the average probability distribution across LFM frequency bands when trained with different

Table 5: Efficiency comparison (in seconds) on Beauty and ML-1M. Training time is measured per epoch; evaluation time is measured for all test users.

Dataset	Beauty		ML-1M	
Model	Train	Eval	Train	Eval
DuoRec	23.3	0.18	161.2	0.17
BSARec	13.5	0.49	85.0	0.19
SLIME4Rec	31.7	0.16	253.6	0.13
MUFFIN (ours)	15.4	0.67	131.5	0.21

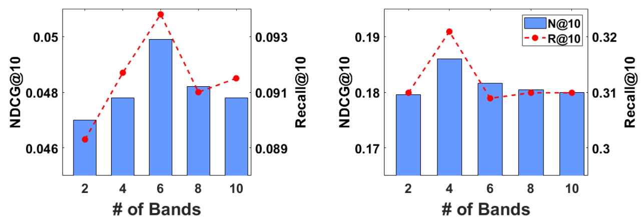
β values on Beauty and ML-1M. The x-axis represents frequency band outputs $\mathbf{o}_t$ in Eq. (19) ordered from the lowest frequency band output to the highest, while the y-axis indicates the corresponding probabilities. We use six and four frequency bands in Beauty and ML-1M, respectively, based on their optimal configurations (see Section 5.4). For Beauty, as shown in Figure 4(a), the probability distribution is well balanced across bands $\mathbf{o}_2$ to $\mathbf{o}_6$ at $\beta = 0.2$ (optimal hyperparameter), demonstrating that MUFFIN effectively mitigates the limited frequency band coverage issue by leveraging a diverse range of frequency components. In contrast, without load balancing loss ($\beta = 0$), LFM heavily concentrates on $\mathbf{o}_1$, *i.e.*, specific low-frequency ranges, thereby leading to the loss of crucial behavioral information. For ML-1M, Figure 4(b) shows similar trends. Notably, low-frequency bands maintain consistently high probabilities regardless of β, reflecting their fundamental role in capturing stable user preferences. Meanwhile, the balanced utilization of high-frequency bands at the optimal β facilitates personalized modeling of diverse behavioral variations across both datasets.

Training and evaluation cost. We analyze the computational efficiency of MUFFIN compared to competing models. As shown in Table 5, MUFFIN demonstrates superior training efficiency compared to the strongest baseline SLIME4Rec [1], achieving approximately 2× and 1.6× faster training time on Beauty and ML-1M, respectively. While SLIME4Rec incurs substantial training overhead due to complicated augmentation and loss computations for contrastive learning, MUFFIN saves significant training time. For evaluation time, MUFFIN shows increased costs compared to baselines, with total evaluation times of 0.67s and 0.21s on Beauty and ML-1M, respectively, compared to the fastest baseline (*i.e.*, SLIME4Rec) times of 0.16s and 0.13s. When calculated per user, this is approximately 0.03-0.035ms of additional time, which stems from the computational overhead of the dual filtering architecture. The increase in evaluation time reflects the trade-off between computational efficiency and the accuracy improvements.

5.4 Hyper-parameter Sensitivity

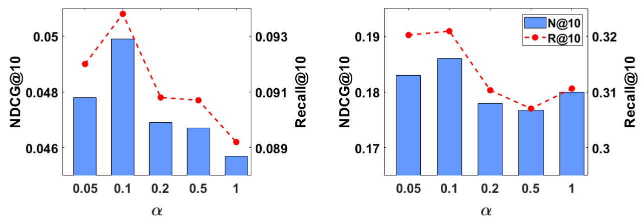
We evaluate the effect of three key hyperparameters used in MUFFIN on model accuracy. For hyperparameter analysis, experiments are conducted with a single seed.

Number of frequency bands K. In Figure 5, we analyzed performance while varying the number of frequency bands (K). When $K = 1$, LFM is equivalent to GFM. Performance improved with increasing K, peaking at $K = 4, 6$. Both too few or too many frequency bands led to suboptimal performance, as insufficient bands limit



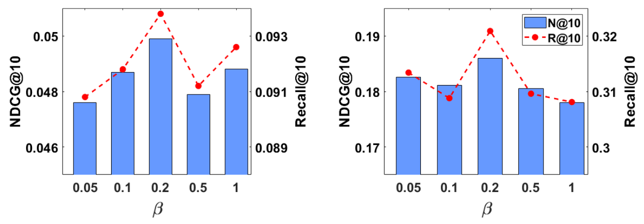
(a) Beauty (b) ML-1M

Figure 5: Accuracy of MUFFIN over varying the number of frequency bands on Beauty and ML-1M.



(a) Beauty (b) ML-1M

Figure 6: Accuracy of MUFFIN over varying auxiliary loss weight α on Beauty and ML-1M.



(a) Beauty (b) ML-1M

Figure 7: Accuracy of MUFFIN over varying load balancing loss weight β on Beauty and ML-1M.

fine-grained pattern capture while excessive bands may cause the overfitting problem.

Auxiliary loss weight α. The weight α helps ensure balanced training between the two modules. We conducted a sensitivity analysis by varying α over a range of values: [0.05, 0.1, 0.2, 0.5, 1], as shown in Figure 6(a) and (b) for the Beauty dataset. Excessively small ($\alpha < 0.05$) or large ($\alpha > 0.5$) values led to performance degradation. MUFFIN exhibited stable and high performance when α was set within the range of 0.05 to 0.1 for most datasets. This sweet spot allows the auxiliary loss to provide sufficient guidance without overwhelming the primary objective.

Load balancing loss weight β. The weight β controls the balance across frequency bands. Figure 7(a) and (b) show its impact when varied from [0.05, 0.1, 0.2, 0.5, 1]. Too small $\beta < 0.05$ causes probability concentration on specific bands, while too large $\beta > 1$ degrades performance by forcing uniform distribution across frequency bands. Most datasets showed the best performance at $\beta = 0.2 \sim 1$. This range ensures that MUFFIN utilizes diverse frequency components while focusing on the informative bands for each user.

6 Related Work

This section reviews existing SR models into two categories: *time-domain models* and *frequency-domain models*.

Time-domain models. Early SR models relied on heuristic-based approaches such as K-nearest neighbor (KNN) methods [7, 13], which were limited in modeling complex, long-range dependencies due to their reliance on simple interaction co-occurrences. The emergence of deep learning [10] significantly advanced the field, enabling more sophisticated modeling of user behavior through neural architectures. Various neural encoders have been explored, including Convolutional Neural Networks (CNNs) [33], Recurrent Neural Networks (RNNs) [11, 18], Graph Neural Networks (GNNs) [8, 39], and Transformers [14, 32, 34]. Among them, SASRec [14] introduced self-attention mechanisms for modeling item-item dependencies, while BERT4Rec [32] leveraged bidirectional Transformers to enhance contextual understanding. In parallel, contrastive learning has emerged as a powerful tool for improving sequence representation learning. For instance, CL4SRec [41] utilized data augmentation strategies to create robust contrastive pairs, and DuoRec [24] introduced supervised augmentation tasks to enrich training signals. Despite these advancements, time-domain models often struggle to capture periodic patterns common in user behavior sequences.

Frequency-domain models. Recent research has turned to the frequency domain for sequence modeling. FMLP-Rec [48] first introduced frequency-based filtering using MLPs to uncover periodic patterns in user-item interactions. Building on this idea, SLIME4Rec [1] and FEARec [2] proposed advanced architectures with layered frequency ramp structures and frequency-aware attention mechanisms. Additionally, some studies [42, 45] have incorporated Fourier-based data augmentation for contrastive learning, further enhancing representation robustness. BSARec [29] introduced fine-grained frequency adjustment as an inductive bias in self-attention layers, aiming to better capture subtle sequential signals. Most recently, some studies have also explored frequency filtering to extract users' distinctive information or filter out noisy components [15, 43]. While these frequency-domain models lack user-specific adaptability across users, MUFFIN dynamically identifies diverse behavioral patterns by using dual complementary modules with user-adaptive filters.

7 Conclusion

In this paper, we present **MUFFIN**, a novel SR model to overcome two major limitations of existing frequency-domain models: limited frequency band coverage and lack of personalized frequency filtering. MUFFIN adopts a dual filtering architecture, comprising GFM and LFM, to effectively exploit the full frequency spectrum. The UAF also enables user-specific frequency filtering for both filtering modules. Extensive experiments demonstrate that MUFFIN consistently outperforms state-of-the-art baselines. Our in-depth analysis confirms that MUFFIN exhibits variability in filters across users. Future work will explore multi-domain scenarios that exhibit more dynamic user behavior patterns and richer frequency characteristics.

References

[1] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Junhua Fang, Guanfeng Liu, Yanchi Liu, Victor S. Sheng, and Xiaofang Zhou. 2023. Contrastive Enhanced Slide Filter

Mixer for Sequential Recommendation. In *ICDE*. 2673–2685.

[2] Xinyu Du, Huanhuan Yuan, Pengpeng Zhao, Jianfeng Qu, Fuzhen Zhuang, Guanfeng Liu, Yanchi Liu, and Victor S. Sheng. 2023. Frequency Enhanced Hybrid Attention Network for Sequential Recommendation. In *SIGIR*. 78–88.

[3] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39, 1 (2020), 1–42.

[4] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Trans. Inf. Syst.* 39, 1 (2020), 10:1–10:42.

[5] Dennis Gabor. 1946. Theory of communication. Part 1: The analysis of information. *Journal of the Institution of Electrical Engineers-part III: radio and communication engineering* 93, 26 (1946), 429–441.

[6] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Transactions on Recommender Systems* 1, 1 (2023), 1–51.

[7] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. 2019. Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN. In *SIGIR*. 1069–1072.

[8] Priyanka Gupta, Diksha Garg, Pankaj Malhotra, Lovekesh Vig, and Gautam M. Shroff. 2019. NISER: Normalized Item and Session Representations with Graph Neural Networks. *CoRR* (2019).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[10] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[12] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[13] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys*. 306–310.

[14] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*. 197–206.

[15] Hye-young Kim, Minjin Choi, Sunkyung Lee, Ilwoong Baek, and Jongwuk Lee. 2025. DIFF: Dual Side-Information Filtering and Fusion for Sequential Recommendation. *arXiv preprint arXiv:2505.13974* (2025).

[16] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J Inman. 2021. 1D convolutional neural networks and applications: A survey. *Mechanical systems and signal processing* 151 (2021), 107398.

[18] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*. 1419–1428.

[19] Shiqi Lin, Zhizheng Zhang, Zhipeng Huang, Yan Lu, Cuiling Lan, Peng Chu, Quanzeng You, Jiang Wang, Zicheng Liu, Amey Parulkar, et al. 2023. Deep frequency filtering for domain generalization. In *CVPR*. 11797–11807.

[20] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*. 43–52.

[21] Henri J Nussbaumer and Henri J Nussbaumer. 1982. *The fast Fourier transform*. Springer.

[22] Namuk Park and Songkuk Kim. 2022. How do vision transformers work? *arXiv preprint arXiv:2202.06709* (2022).

[23] Seongmin Park, Mincheol Yoon, Minjin Choi, and Jongwuk Lee. 2025. Temporal Linear Item-Item Model for Sequential Recommendation. In *WSDM*. 354–362.

[24] Ruihong Qiu, Zi Huang, Hongzhi Yin, and Zijian Wang. 2022. Contrastive Learning for Representation Degeneration Problem in Sequential Recommendation. In *WSDM*. 813–823.

[25] Lawrence R Rabiner and Bernard Gold. 1975. *Theory and Application of Digital Signal Processing*. Prentice-Hall.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[27] Ervin Sejdić, Igor Djurović, and LJubiša Stanković. 2011. Fractional Fourier transform as a signal processing tool: An overview of recent developments. *Signal Processing* 91, 6 (2011), 1351–1369.

[28] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538* (2017).

[29] Yehjin Shin, Jeongwhan Choi, Hyowon Wi, and Noseong Park. 2024. An Attentive Inductive Bias for Sequential Recommendation beyond the Self-Attention. In *AAAI*. 8984–8992.

[30] Samir S Soliman and Mandyam D Srinath. 1990. Continuous and discrete signals and systems. *Englewood Cliffs* (1990).

[31] H V Sorensen, D Jones, Michael Heideman, and C Burrus. 1987. Real-valued fast Fourier transform algorithms. *IEEE Transactions on acoustics, speech, and signal processing* 35, 6 (1987), 849–863.

[32] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*. 1441–1450.

[33] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.

[34] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[35] Mengzhu Wang, Jianlong Yuan, and Zhibin Wang. 2023. Mixture-of-experts learner for single long-tailed domain generalization. In *MM*. 290–299.

[36] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*. 6332—-633.

[37] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*. 6332–6338.

[38] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2022. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering* 35, 5 (2022), 4425–4445.

[39] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-Based Recommendation with Graph Neural Networks. In *AAAI*. 346–353.

[40] Yadong Xiao, Jiajin Huang, and Jian Yang. 2024. TFCSRec: Time–frequency consistency based contrastive learning for sequential recommendation. *Expert Systems with Applications* 245 (2024), 123118.

[41] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive Learning for Sequential Recommendation. In *ICDE*. 1259–1273.

[42] Zhibin Yang, Jiwei Qin, Donghao Zhang, Jie Ma, and Peichen Ji. 2024. Adaptive Frequency Domain Data Augmentation for Sequential Recommendation. *IEEE Access* (2024).

[43] Junjie Zhang, Ruobing Xie, Hongyu Lu, Wenqi Sun, Wayne Xin Zhao, Yu Chen, and Zhanhui Kang. 2025. Frequency-Augmented Mixture-of-Heterogeneous-Experts Framework for Sequential Recommendation. In *WWW*. 2596–2605.

[44] Qianru Zhang, Peng Yang, Honggang Wen, Xinzhu Li, Haixin Wang, Fang Sun, Zezheng Song, Zhichen Lai, Rui Ma, Ruihua Han, et al. 2025. FreqMoE: Enhancing Time Series Forecasting through Frequency Decomposition Mixture of Experts. *AISTATS* (2025).

[45] Yichi Zhang, Guisheng Yin, and Yuxin Dong. 2023. Contrastive learning with frequency-domain interest trends for sequential recommendation. In *RecSys*. 141–150.

[46] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2021. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In *CIKM*. 4653–4664.

[47] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*. 1893–1902.

[48] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Filter-enhanced MLP is All You Need for Sequential Recommendation. In *WWW*. 2388–2399.