MLP4Rec: A Pure MLP Architecture for Sequential Recommendations

Muyang Li[1], **Xiangyu Zhao**[2*], **Chuan Lyu**[3], **Minghao Zhao**[4], **Runze Wu**[4], **Ruocheng Guo**[5]
[1]University of Sydney
[2]City University of Hong Kong
[3]Zhejiang University
[4]Fuxi AI Lab, Netease
[5]Bytedance AI Lab
muli0371@uni.sydney.edu.au, xianzhao@cityu.edu.hk, chuan_lyu@zju.edu.cn,
{zhaominghao,wurunze1}@corp.netease.com, rguo.asu@gmail.com

Abstract

Self-attention models have achieved state-of-the-art performance in sequential recommender systems by capturing the sequential dependencies among user-item interactions. However, they rely on positional embeddings to retain the sequential information, which may break the semantics of item embeddings. In addition, most existing works assume that such sequential dependencies exist solely in the item embeddings, but neglect their existence among the item features. In this work, we propose a novel sequential recommender system (MLP4Rec) based on the recent advances of MLP-based architectures, which is naturally sensitive to the order of items in a sequence. To be specific, we develop a tri-directional fusion scheme to coherently capture sequential, cross-channel, and cross-feature correlations. Extensive experiments demonstrate the effectiveness of MLP4Rec over various representative baselines upon two benchmark datasets. The simple architecture of MLP4Rec also leads to linear computational complexity as well as much fewer model parameters than existing self-attention methods.

1 Introduction

Accurately modeling the chronological behavior of users is a critical area of research in recommender systems. The primary challenge is to capture the sequential pattern of user interests across multiple items, which is typically dynamic. To address this issue, Sequential Recommender Systems (SRS) were proposed and have garnered considerable interests from both academia and industry. While many endeavors have been put into this field, the newly emerged self-attention mechanism [Vaswani *et al.*, 2017] has gained a dominant position in SRS. Recent works show that self-attention based models can significantly outperform other models, and have achieved state-of-the-art (SOTA) performances in SRS [Kang and McAuley, 2018; Zhang *et al.*, 2019; Sun *et al.*, 2019].

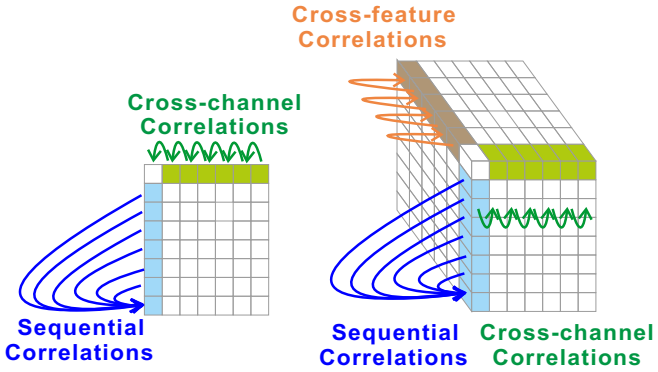

Figure 1: Bi-directional correlations v.s. Tri-directional correlations

Despite the success of self-attention in sequential recommendations, some limitations can potentially restrict its further development and practical applications. First, self-attention and its cognate methods are insensitive to the sequential order of the input items, and therefore relies on extra process such as adding positional embeddings to the input sequence to make the model aware of the information contained in the order of sequence. However, existing self-attention methods, combining item sequence and positional embeddings from two heterogeneous data types, may interrupt the underlying semantics of item embeddings [Zheng *et al.*, 2021]. Second, self-attention methods' computational complexity is quadratic to the length of the input item sequence, which yields an unneglectable computational cost for large-scale recommender systems. Third, incorporating self-attention in recommender systems typically leads to huge amounts of model parameters, which result in difficulty in model optimization and an increased chance of over-fitting.

Recent advances in Multi-layer Perceptron (MLP) architectures, such as MLP-Mixer, gMLP and resMLP [Tolstikhin *et al.*, 2021; Liu *et al.*, 2021; Touvron *et al.*, 2021], show competitive performances in computer vision tasks despite their architectural simplicity and linear computational complexity. This questions the necessity of attention mechanisms and shows the possibility to replace them via simple MLP architectures. To address aforementioned challenges of

*Xiangyu Zhao is corresponding author.

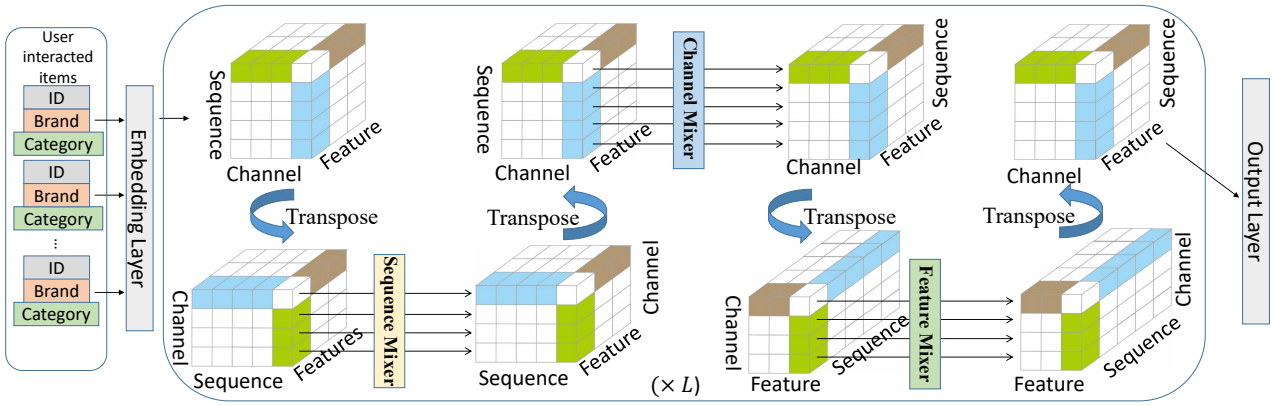

Figure 2: Overall framework of MLP4Rec.

self-attention based SRS, this paper proposes a simple yet effective **MLP** framework for sequential **Rec**ommendations (**MLP4Rec**), which has two-fold advantages. First, along with the above MLP-based models, MLP4Rec is by design sensitive to the order of input item sequence, avoiding the bottleneck caused by using positional embeddings. Second, upon pure MLP blocks, MLP4Rec possesses linear computational complexity and a significantly lower amount of model parameters than self-attention based SRS models.

However, due to the design of bi-directional mixer in existing MLP architectures [Tolstikhin *et al.*, 2021], utilizing them for sequential recommendations can only capture the dependencies of item embeddings and incorporate the items' explicit features (e.g., brand and category) in a naive manner. To this end, we devise a novel tri-directional information fusion scheme for MLP4Rec with a cross-feature mixer, which enables the framework to capture the interactions among all item features, as illustrated in Figure 1. In addition, the tri-directional scheme also applies the classic bi-directional mixers from MLP-based models [Tolstikhin *et al.*, 2021; Lee *et al.*, 2021] on item explicit features, which learns the users' sequential preferences within these features. Through extensive experiments, we demonstrate that MLP4Rec shows significantly superior performance than the state-of-the-art methods on two benchmark datasets. To summarize, this paper has the following contributions:

(1) We investigate the possibility of replacing the self-attention mechanism with simple MLP architectures for sequential recommendations;

(2) To the best of our knowledge, this is the first work that proposes a tri-directional mixing MLP architecture;

(3) We validate the effectiveness of our proposed framework via extensive experiments on two benchmark datasets.

2 Framework

In this section, we discuss the framework, methodology, and optimization of our proposed MLP4Rec framework.

2.1 Notation Definition

Follow commonly adapted settings [Li *et al.*, 2018; Kang and McAuley, 2018], we denote the participant of interactions - users as $U = \{u_1, ..., u_n..., u_N\}$, where n indicates the ID of the user. Items as $I = \{i_1, ..., i_m..., i_M\}$, where m indicates the ID of the item. In addition, each item have some associated features, such as category and brand, we denote those features as $Q = \{q_1^m, ..., q_k^m, ...q_K^m\}$, where q_k^m refers to the k-th feature of item m. We sort the items that users have interacted with into sequences, thus each user has a corresponding sequence containing items (s)he once viewed chronologically. We denote the item sequence of user n as $S_n = \{i_1, ..., i_t..., i_s\}$, where i stands for item, t describes the chronological order of item, s is the maximum length of the sequence.

2.2 Framework Overview

Here, we present our MLP-based SRS which can explicitly learn tri-directional information. As we mentioned before, in order to make an informed prediction, a model must be able to capture the 3-fold information. The first fold is the temporal information, i.e., sequential dependencies among S_n. The second fold refers to, the interest information contained in the item embedding, since different channels (dimensions) of an item embedding represents different latent semantics, the cross-channel correlation is also important for our task. The third fold is the correlations among item features, collectively, they contribute to modeling the semantic meaning of an item. By repetitively transposing and applying MLP blocks in different directions of the input embedding tensor as shown in Figure 2, we show that our proposed framework can capture the sequential, cross-channel, and cross-feature correlations at the same time.

To be specific, MLP4Rec consists of L layers, where each layer has an identical setting, a sequence-mixer, a channel-mixer, and a feature-mixer. Following [Tolstikhin *et al.*, 2021], all L layers share the same parameters to reduce model parameters. Within each layer, we first apply independent

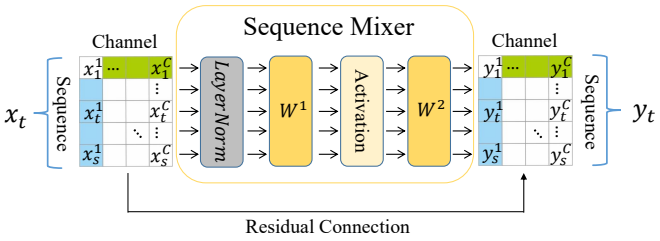

Figure 3: Architecture of Sequence-Mixer

sequence-mixers and channel-mixers for different features, so as to learn their unique characteristics. Then, we utilize a feature-mixer to learn correlations among all features.

2.3 Detailed Architecture

Embedding Layer. We adapt a commonly used method for constructing item ID embeddings and feature embeddings, i.e., learning an embedding lookup table to project the discrete item identifiers (i.e., IDs) and explicit features (e.g., category and brand) into dense vector representations with dimension C [Cheng *et al.*, 2016]. After the embedding layer, we can stack the embeddings of item IDs and explicit features into individual embedding tables, where the row of the embedding table is each embedding vector, and the column of the embedding table contains channel information. Stacking all embedding tables together, we obtain a 3-d embedding table as shown in Figure 3. Note that, unlike self-attention models, our proposed model does not need to learn a positional embedding for an input sequence. Instead, temporal information can be directly learned via the sequence-mixer.

Sequence-Mixer. The sequence-mixer is an MLP block, which aims to learn the sequential dependencies across the entire item sequence. The sequence-mixer block takes the rows of the embedding table as input features (applied to the transposed embedding table), and outputs an embedding table with the same dimension as the input. But in this output table, all the sequential dependencies are fused within each output sequence. More specifically, a set of input feature would be the c-th dimension of each embedding vector across the whole sequence, i.e. $\{x_1^c, ..., x_t^c, ..., x_s^c\}$ as shown in Figure 3. The correlation between them is sequential, which shows the evolvement of user interest across time, thus making the sequence-mixer sensitive to the sequential order. Formally, we denote the output of sequence-mixer at layer l as:

$$\boldsymbol{y_t} = \boldsymbol{x_t} + \boldsymbol{W^2} g^l(\boldsymbol{W^1} LayerNorm(\boldsymbol{x_t})) \quad (1)$$

where t = 1,..,s. $\boldsymbol{x_t}$ is the input feature, which is the embedding vector at time step t. $\boldsymbol{y_t}$ is the output of the block, g^l is the non-linear activation function at layer l, $\boldsymbol{W^1} \in \mathbb{R}^{r_s \times s}$ denotes the learnable weights representing the first fully connected layer in the sequence-mixer, $\boldsymbol{W^2} \in \mathbb{R}^{s \times r_s}$ signifies the learnable weights of the second fully connected layer in the sequence-mixer, r_s is the tunable hidden size of sequence-mixer. We employ layer normalization (LayerNorm) [Ba *et al.*, 2016] and residual connection [He *et al.*, 2016] as in MLP-mixer [Tolstikhin *et al.*, 2021].

Channel-Mixer. Like sequence-mixer, channel-mixer is also an MLP block with a similar macro architecture, their key distinction is between their purpose. The objective of the channel-mixer is to learn the correlation within an embedding vector. The embedding of an item ID or item feature usually expresses some latent semantics on each dimension, learning their representation and internal correlation is crucial for recommendations. The channel-mixer takes the columns of the embedding table as input feature, as shown in Figure 2, channel-mixer is applied after transposing the embedding table back to its original shape. After the sequence-mixer, sequential information is fused within each sequence, but the cross-channel correlation has not been discovered yet. Channel-mixer will take t-th item embedding's dimension as input feature, i.e. $\{x_t^1, ..., x_t^c, ..., x_t^C\}$, the correlation between them is cross-channel, collectively they express the overall semantic of the embedding. After the channel-mixer, the cross-channel correlation will be fused into the output sequence. We denote the output of channel-mixer at layer l as:

$$\boldsymbol{y_c} = \boldsymbol{x_c} + \boldsymbol{W^4} g^l(\boldsymbol{W^3} LayerNorm(\boldsymbol{x_c})) \quad (2)$$

where c = 1,2,..,C, $\boldsymbol{x_c}$ is the input feature, which is the cth dimension across all embedding at time step t, and $\boldsymbol{y_c}$ is the output of the block,$\boldsymbol{W^3} \in \mathbb{R}^{r_C \times C}$ is learnable weights of the first fully connected layer in the channel-mixer, $\boldsymbol{W^4} \in \mathbb{R}^{C \times r_C}$ is learnable weights of the second fully connected layer, r_C is tunable hidden size in channel-mixer.

Feature-Mixer. After the sequence-mixer and the channel-mixer, the sequential and cross-channel dependencies are fused within each sequence. However, the information among the embedding table of different features is still independent of each other. The feature-mixer can fuse cross-feature correlation into the representation of each sequence. More importantly, since the feature-mixer is the last block in a layer, which not only communicates feature information but also shares the sequential and cross-channel dependencies within each feature to other features, thus coherently connects the tri-directional information. The feature-mixer acts on features dimension as shown in Figure 2. We denote the output of the feature-mixer at layer l as:

$$\boldsymbol{y_k} = \boldsymbol{x_k} + \boldsymbol{W^6} g^l(\boldsymbol{W^5} LayerNorm(\boldsymbol{x_k})) \quad (3)$$

where k = 1,2,...,K, $\boldsymbol{x_k}$ is the input feature, which is the embedding vector of kth feature at embedding dimension c, and $\boldsymbol{y_k}$ is the output of the block, $\boldsymbol{W^5} \in \mathbb{R}^{r_K \times K}$ denotes the learnable weights of the first fully connected layer in the feature-mixer, $\boldsymbol{W^6} \in \mathbb{R}^{K \times r_K}$ is the learnable weights of the second fully connected layer in the feature-mixer, and r_K is tunable hidden size in feature-mixer.

2.4 Training and Inference

Training. We adapt Cross-Entropy loss as the loss function for our model:

$$L = -\sum_{S_n \in S} \sum_{t \in [1,...,s]} [log(\sigma(r_{i_t,t})) + \sum_{j \notin S_n} log(1 - \sigma(r_{i_j,t}))] \quad (4)$$

where σ demotes sigmoid function, $r_{i_t,t}$ is model's predicted similarity to ground-truth item i_t, and $r_{i_j,t}$ is the predicted

similarity to sampled items at timestep t, j is the negative sampled items, S is the set of all users' interaction sequences.

Inference. We adapt the most commonly used inference method in SRS for fair comparison [Kang and McAuley, 2018; Zhang *et al.*, 2019]. To be specific, after L layers of sequence-mixer, channel-mixer and feature-mixer, we obtain a sequence of hidden states that contains the sequential, cross-channel, and cross-feature dependencies of each interaction, respectively. Assuming at time step t, we wish to predict next item i_{t+1}, given sequence of hidden states $H = h_1, ..., h_t$, we can calculate the cosine similarity between h_t and all candidates items E_m via dot product as:

$$r_{m,t} = h_t \cdot E_m^T \tag{5}$$

where m = 1,...,M, $E_m \in \mathbb{R}^{M \times C}$ is the item embedding of all candidate items and $r_{t,m}$ indicates the similarity between hidden state t to all candidate items, top predictions will be ranked by their similarity.

2.5 Discussion

Relation to MLP-Mixer and resMLP. The key architectural differences of MLP4Rec to MLP-Mixer and resMLP is that MLP-Mixer and resMLP directly project a 3-dimensional input (image) into a 2-dimensional embedding table, and then operate 2-dimensional (spatial/channel) information fusion, whereas MLP4Rec directly operates on the 3-dimensional input and conducts the sequential/channel/feature information fusion. MLP4Rec can degenerate into MLP-Mixer and resMLP when the input is a 2-dimensional embedding table.

Complexity Analysis. The following discussion regarding the time and space complexity of our model is for the inference stage. (1) Time Complexity: MLP4Rec's time complexity is $O(s + C + K)$, which is linear complexity to the sequence length s, embedding size C and feature number K. Compared to the time complexity of self-attention, $O(s^2C + C^2s)$, the theoretical upper bound of the MLP4Rec's time complexity is significantly lower. (2) Space Complexity: MLP4Rec's space complexity is $O(K(s + C + 1))$, where the number of features K is usually limited, especially after feature selection. On the other hand, the space complexity of self-attention is $O(sC + C^2)$ [Kang and McAuley, 2018], which is quadratic to the embedding size. In the experiment part, we show that not only do we keep a theoretical lower upper bound in space complexity, but in practice, we also achieved a significantly lower number of parameters.

3 Experiments

This section evaluates the performance of MLP4Rec against representative baselines on two benchmark datasets.

3.1 Datasets

We choose two widely used datasets to benchmark our performance on both small and large datasets, and their statistics can be found in Table 1. (1) **MovieLens**[1]: MovieLens is a site for recommending movies to users given their historical ratings, which is now one of the most commonly used benchmarks across the field of recommender system. We use MovieLens-100k in our experiments. (2) **Amazon Beauty**[2]: The online reviews and ratings of Amazon. We use the "Beauty" category in our experiments. We filter out the items and users that have less than 5 interactions for two datasets. We set the maximum sequence length as 50 for both datasets, and conduct zero-padding for shorter sequences.

Data	MovieLens	Beauty
# interactions	100,000	2,023,070
# users	943	1,210,271
# items	1,682	249,274
# avg. length	106	8.8

Table 1: Statistics of the datasets.

3.2 Evaluation Settings

We employ the commonly used evaluation method in SRS, namely next-item prediction. For dataset splitting, the next-item prediction task uses the last item in an interaction sequence as the test set, the item before as the validation set, and the rest of the items will be used as the training set. Following common settings, we pair 100 negative samples with ground-truth items during prediction [Kang and McAuley, 2018].

Metrics. We apply three commonly used evaluation metrics in the recommendations, namely Hit Ratio (HR), Normalized Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). All results are averaged on three random seeds.

3.3 Implementation Details

The implementation of MLP4Rec and all baselines are based on RecBole's library [Zhao *et al.*, 2021], an open-source recommender system library, which allows us to test and compare all methods in a fair environment, and allows our results to be reproduced easily. We tune the hyper-parameters based on original papers' recommendations. If original papers did not provide detailed hyper-parameters, we perform hyper-parameter tuning via cross-validation with Adam optimizer [Kingma and Ba, 2014] and early stop strategy. The implementation code is available online[3].

3.4 Performance Comparison

We will compare our proposed methods against following baselines: PopRec, BPR [Rendle *et al.*, 2009], FPMC [Rendle *et al.*, 2010], GRU4Rec [Hidasi *et al.*, 2015], GRU4Rec$^+$ [Hidasi *et al.*, 2016], SASRec and SASRec$^+$ [Kang and McAuley, 2018], BERT4Rec [Sun *et al.*, 2019], FDSA [Zhang *et al.*, 2019], and MLP-Mixer$^+$ [Tolstikhin *et al.*, 2021]. Note that superscript "+" means that we improve the original model, which takes the concatenation of embeddings of item ID and features as input, enabling fair comparison with MLP4Rec.

Table 2 summarizes the comparison results, where models above the dashed line consider only item embeddings, while below models also involve item features. From Table 2, we can make the following general observations: (1)

[1]https://grouplens.org/datasets/movielens/100k/

[2]http://jmcauley.ucsd.edu/data/amazon/

[3]https://github.com/Li-Muyang/MLP4Rec

Methods	MovieLens			Beauty		
Metrics	MRR@10	NDCG@10	HR@10	MRR@10	NDCG@10	HR@10
PopRec	0.1496	0.0783	0.4044	0.0305	0.0443	0.0954
BPR	0.1479	0.1905	0.3474	0.1479	0.3474	0.1905
FPMC	0.1220	0.1813	0.3810	0.1431	0.1838	0.3165
GRU4Rec	0.1860	0.2550	0.4758	0.1632	0.2050	0.3417
SASRec	0.1901	0.2612	0.4920	0.2009	0.2447	0.3874
BERT4Rec	0.1819	0.2568	0.5061	0.1313	0.1738	0.3135
GRU4Rec$^+$	0.1880	0.2550	0.4758	0.1848	0.2294	0.3746
SASRec$^+$	0.2022	0.2710	0.4970	0.2045	0.2488	0.3930
FDSA	0.1913	0.2625	0.4984	0.2056	0.2522	0.4040
MLP-Mixer$^+$	0.1987	0.2671	0.4920	0.2089	0.2556	0.4065
MLP4Rec	**0.2027**	**0.2747**	**0.5118**	**0.2139***	**0.2654***	**0.4326***

"*" indicates the statistically significant improvements (i.e., two-sided t-test with $p < 0.05$) over the best baseline.

Table 2: Overall performance comparison on two datasets, where best baseline performances are underlined

Model	Param	NDCG@10	HR@10
BERT4Rec	2.3M	0.1738	0.3135
GRU4Rec$^+$	2.5M	0.2294	0.3746
SASRec$^+$	2.0M	0.2488	0.3930
FDSA	3.2M	0.2522	0.4040
MLP-Mixer$^+$	1.8M	0.2556	0.4065
MLP4Rec	**1.7M**	**0.2654**	**0.4326**

"Param" refers to the number of trainable model parameters.

Table 3: Model parameter analysis on Beauty dataset.

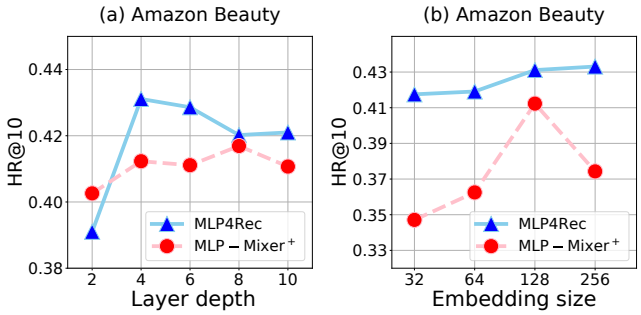

Figure 4: Influence of hyper parameters on performance

Starting from GRU4Rec, deep learning based methods exceed traditional methods such as BPR by a large margin, suggesting that in sequential recommendations, deep learning models are better at capturing sequential dependencies. (2) Models that can handle item features (e.g. SASRec$^+$, FDSA) usually outperform those who cannot (e.g. SASRec, BERT4Rec), indicating the importance of item features in sequential recommendations. (3) Improvement over the best baseline is more significant on the larger dataset "Beauty". More specifically, we can also observe that: (4) Compared to RNN-based models, self-attention models usually have better performances, which can be attributed to self-attention's stronger capabilities in capturing sequential patterns. (5) MLP-Mixer$^+$ can achieve comparable performance when compared with the SOTA methods such as SASRec and FDSA. (6) MLP4Rec constantly outperforms all baselines including MLP-Mixer$^+$ with a significant margin, which suggests that tri-directional information fusion is an important improvement, which jointly captures sequential, cross-channel, cross-feature correlations.

3.5 Model Parameter Analysis

As shown in Table 3, despite MLP4Rec's superior performance, it also surpasses baselines in terms of memory efficiency. Fewer model parameters not only make the MLP4Rec easier to train, but also reduce the risk of over-fitting [Lee *et al.*, 2021].

3.6 Hyper-parameters Analysis

Figure 4 shows the influence of layer depth and embedding size on MLP4Rec and MLP-Mixer$^+$. Generally, unlike MLP-Mixer's application in CV [Tolstikhin *et al.*, 2021], our framework in SRS does not require a very deep network. In addition, compared to MLP-Mixer$^+$, the performance of MLP4Rec is more robust across a wide range of embedding sizes. A potential reason for this is that the tri-directional information communication allows latent representations to be shared on different embedding tables, thus a smaller embedding size does not significantly harm the representational capacity of the model. However, MLP-Mixer$^+$ needs to compress rich semantics from item features. Thus, small embedding sizes lead to sub-optimal performance due to their limited representational ability. In contrast, a large embedding size results in an over-fitting issue.

3.7 Ablation Study

As shown in the previous subsections, MLP4Rec achieves better performance than MLP-Mixer$^+$ in both datasets across all metrics, and the only difference between their architecture is feature-mixer. Here, we investigate the necessity of a feature-mixer by answering two important questions: *Q1:*

Model	MRR@10	NDCG@10	HR@10
MLP-Mixer	0.1974	0.2401	0.3790
MLP4Rec-Linear	0.2100	0.2586	0.4165
MLP4Rec-Simple	0.1995	0.2500	0.4143
MLP-Mixer$^+$	0.2089	0.2522	0.4040
w/o S-mixer	0.1771	0.2153	0.3396
w/o C-mixer	0.1933	0.2440	0.4092
w/o F-mixer	0.1974	0.2401	0.3790
MLP4Rec	**0.2139**	**0.2654**	**0.4326**

Table 4: Ablation study comparison

Can simpler alternatives achieve the same performance as the feature-mixer? and *Q2: What is the contribution of each respective module in our proposed model?* To answer those questions, and further validate the importance of our proposed improvement, we design the following alternatives to MLP4Rec and MLP-Mixer: (1) **MLP-Mixer** is the vanilla MLP-Mixer which does not include item features. (2) **MLP4Rec-Linear** is the simplified MLP4Rec which replaces the MLP feature-mixer block with a simple linear layer. (3) **MLP4Rec-Simple** is the simplified MLP4Rec which only performs feature mixing at the final layer instead of every layer. (4) **w/o S-Mixer** is the simplified MLP4Rec without the Sequence-Mixer module. (5) **w/o C-Mixer** is the simplified MLP4Rec without the Channel-Mixer module. (6) **w/o F-Mixer** is the simplified MLP4Rec without the Feature-Mixer module.

The performances of alternatives are displayed in Table 4, where the upper part of the table mainly addresses *Q1* and the lower part of the table mainly addresses *Q2*. From Table 4, we can summarize that (1) Without incorporating item features, MLP-Mixer has significantly worse performance, which confirmed the importance of introducing item features into sequential recommendations. (2) For two simplified MLP4Rec versions, we can observe that MLP4Rec-Linear constantly outperforms MLP4Rec-Simple, which means that only communicating feature information at the last layer cannot fuse cross-feature correlation into the hidden representation sufficiently. Whereas MLP4Rec-Linear still performs proper tri-directional information without nonlinearity, leading to the next best performance other than MLP4Rec. (3) MLP4Rec outperforms MLP4Rec-Linear and MLP4Rec-Simple consistently over all metrics, attributing to its full tri-directional fusion by feature-mixer. (4) Without Sequence-Mixer, MLP4Rec's performance degenerates most significantly, indicating that Sequence-Mixer plays a vital part in the sequential recommendations, and can successfully capture the sequential pattern. (5) Without Channel-Mixer, MLP4Rec also suffers a significant decrease in performance. The most likely reason is that, without the Channel-Mixer, the respective dimension of item/feature embedding cannot communicate with one another, thus making the hidden representation lack cross-channel correlation. And since without Feature-Mixer, the performance of MLP4Rec is essentially equivalent to vanilla MLP-Mixer, so the effect of removing Feature-Mixer can refer to summarization (1).

4 Related Work

In this section, we review the related work from the literature of sequential recommendation systems and MLP-Mixer.

Sequential Recommendation Systems. RNN-based models can handle complex sequential dependencies in sequential recommendations by compressing previous user-item interactions into a vector that summarizes that information, and then make the prediction of the next possible interaction [Quadrana *et al.*, 2017; Yu *et al.*, 2016; Zhao *et al.*, 2018a; Zhao *et al.*, 2018b; Zhao *et al.*, 2019]. For example, GRU4Rec [Hidasi *et al.*, 2015] is one of the most representative RNN-based SRS, which implements gated recurrent unit (GRU) to improve the modeling of long-term dependencies, however, even with GRU, RNN-based models still cannot perform very well on a long sequence.

Recent years, (self-)attention methods [Vaswani *et al.*, 2017; Li *et al.*, 2017] show SOTA performances in SRS. SASRec [Kang and McAuley, 2018] is one of the first to implement self-attention for SRS and obtains promising results, by stacking several self-attention blocks, SASRec is able to capture complex dependencies among items.

BERT4Rec [Sun *et al.*, 2019] implements bi-directional self-attention blocks and Cloze objective, which also shows promising results. FDSA uses self-attention on both item token and item features to gain more information for better prediction. Nevertheless, self-attention's drawbacks are just as significant, whose computational complexity is quadratic to the length of the input sequence and embedding size.

MLP-based Architectures. Recent development in MLP architectures reveals high potential in computer vision [Tolstikhin *et al.*, 2021; Touvron *et al.*, 2021; Liu *et al.*, 2021]. Among them, MLP-Mixer [Tolstikhin *et al.*, 2021] is a symbolic example of recent advances in MLP-based models. MLP-Mixer uses token-mixer and channel-mixer to separately learn the spatial and channel correlations. With linear computation complexity and simpler architectures, MLP-Mixer was reported to have comparable performance compared with SOTA methods. MOI-Mixer [Lee *et al.*, 2021] is the first work to investigate the possibility of implementing MLP-Mixer in the sequential recommendation. They propose a Multi-Order-Interaction layer to improve vanilla MLP-Mixer's performance.

5 Conclusion

In this paper, we proposed a simple but efficient architecture with only MLP blocks for sequential recommendations. This architecture leverages a novel way to coherently connects sequential, cross-channel and cross-feature correlations in users' historical interaction data to mine their preference. MLP4Rec shows superior performances against state-of-the-art methods with a significant margin on two commonly used benchmark datasets, validating that: (1) MLP4Rec offers a powerful alternative to current self-attention based methods; (2) Feature-mixer enables the proposed model to cope with heterogeneous features and capture their correlations. In addition, MLP4Rec's simpler model architecture and much fewer model parameters enhance its scalability in large-scale practical recommender systems.

Acknowledgments

This research was partially supported by the APRC - CityU New Research Initiatives (No.9610565, Start-up Grant for the New Faculty of the City University of Hong Kong), the SIRG - CityU Strategic Interdisciplinary Research Grant (No.7020046, No.7020074), the HKIDS Early Career Research Grant, and the CCF-Tencent Open Fund.

References

[Ba *et al.*, 2016] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[Cheng *et al.*, 2016] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proc. of DLRS*, 2016.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, 2016.

[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.

[Hidasi *et al.*, 2016] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proc. of RecSys*, 2016.

[Kang and McAuley, 2018] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *Proc. of ICDM*, 2018.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Lee *et al.*, 2021] Hojoon Lee, Dongyoon Hwang, Sunghwan Hong, Changyeon Kim, Seungryong Kim, and Jaegul Choo. Moi-mixer: Improving mlp-mixer with multi order interactions in sequential recommendation. *arXiv preprint arXiv:2108.07505*, 2021.

[Li *et al.*, 2017] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proc. of CIKM*, 2017.

[Li *et al.*, 2018] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proc. of KDD*, 2018.

[Liu *et al.*, 2021] Hanxiao Liu, Zihang Dai, David R So, and Quoc V Le. Pay attention to mlps. *arXiv preprint arXiv:2105.08050*, 2021.

[Quadrana *et al.*, 2017] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proc. of RecSys*, 2017.

[Rendle *et al.*, 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. of UAI*, 2009.

[Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proc. of WWW*, 2010.

[Sun *et al.*, 2019] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proc. of CIKM*, 2019.

[Tolstikhin *et al.*, 2021] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. In *Proc. of NeurIPS*, 2021.

[Touvron *et al.*, 2021] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv preprint arXiv:2105.03404*, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. of NeurIPS*, 2017.

[Yu *et al.*, 2016] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *Proc. of SIGIR*, 2016.

[Zhang *et al.*, 2019] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *Proc. of IJCAI*, 2019.

[Zhao *et al.*, 2018a] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proc. of RecSys*, 2018.

[Zhao *et al.*, 2018b] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proc. of KDD*, 2018.

[Zhao *et al.*, 2019] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. Deep reinforcement learning for search, recommendation, and online advertising: a survey. *ACM SIGWEB Newsletter*, 2019.

[Zhao *et al.*, 2021] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *Proc. of CIKM*, 2021.

[Zheng *et al.*, 2021] Jianqiao Zheng, Sameera Ramasinghe, and Simon Lucey. Rethinking positional encoding. *arXiv preprint arXiv:2107.02561*, 2021.