Recurrent Convolutional Neural Network for Sequential Recommendation

Chengfeng Xu
Institute of AI, Soochow University
suzhou, China
cfxu@stu.suda.edu.cn

Pengpeng Zhao*
Institute of AI, Soochow University
suzhou, China
ppzhao@suda.edu.cn

Yanchi Liu
Rutgers University
New Jersey, USA
yanchi.liu@rutgers.edu

Jiajie Xu
Institute of AI, Soochow University
suzhou, China
xujj@suda.edu.cn

Victor S.Sheng
University of Central Arkansas
Conway, Arkansas, USA
ssheng@uca.edu

Zhiming Cui
Suzhou University of Science and Technology, China
zmcui@mail.usts.edu.cn

Xiaofang Zhou
The University of Queensland
Queensland, Australia
zxf@itee.uq.edu.au

Hui Xiong
Rutgers University
New Jersey, USA
hxiong@rutgers.edu

ABSTRACT

The sequential recommendation, which models sequential behavioral patterns among users for the recommendation, plays a critical role in recommender systems. However, the state-of-the-art Recurrent Neural Networks (RNN) solutions rarely consider the non-linear feature interactions and non-monotone short-term sequential patterns, which are essential for user behavior modeling in sparse sequence data. In this paper, we propose a novel **R**ecurrent **C**onvolutional **N**eural **N**etwork model (RCNN). It not only utilizes the recurrent architecture of RNN to capture complex long-term dependencies, but also leverages the convolutional operation of Convolutional Neural Network (CNN) model to extract short-term sequential patterns among recurrent hidden states. Specifically, we first generate a hidden state at each time step with the recurrent layer. Then the recent hidden states are regarded as an "image", and RCNN searches non-linear feature interactions and non-monotone local patterns via intra-step horizontal and inter-step vertical convolutional filters, respectively. Moreover, the output of convolutional filters and the hidden state are concatenated and fed into a fully-connected layer to generate the recommendation. Finally, we evaluate the proposed model using four real-world datasets from various application scenarios. The experimental results show that our model RCNN significantly outperforms the state-of-the-art approaches on sequential recommendation.

KEYWORDS

Sequential Recommendation; Recurrent Neural Network; Convolutional Neural Network.

*The corresponding author

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313408

ACM Reference Format:
Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S.Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent Convolutional Neural Network for Sequential Recommendation. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313408

1 INTRODUCTION

Recommender systems play a significant role in engaging and satisfying users with services provided in the age of information explosion. To build effective recommender systems, a key factor is to accurately characterize users' interests and tastes, which are intrinsically dynamic and evolving. To achieve this goal, the sequential recommendation has been proposed to recommend the successive item(s) that a user is likely to interact with based on his/her past activity sequences.

Early sequential methods like Markov chains [18] are proposed for modeling user's sequential pattern. However, these methods based on Markov chains model local sequential behaviors between every two adjacent items but have difficulties to model high-order relationships. More recently, deep neural networks have been intensively studied in related domains and made extraordinary impacts on sequential recommendation. For instance, Recurrent Neural Network (RNN) has become a popular sequential neural model. For better holding the long-term dependencies, two variants of RNN in particular, namely the Long-Short Term Memory (LSTM) [5] and the Gated Recurrent Unit (GRU) [1] have been widely used. Besides, other variants of RNN, including session-based RNN [4], hierarchical personalized RNN [10], and attention-based RNN [8] are found helpful in modeling sequences.

However, RNN holds an assumption that temporal dependency changes monotonically [9], which means that the current item or hidden state is more important than the previous one. This monotonic temporal dependency of RNN impairs the modeling of user's short-term interest. Literature work has shown that feature interactions can help capture sequential patterns especially with

sparse data [3, 11, 21]. Thus, it is crucial to consider the interactions between hidden state features in RNN when learning from sparse sequential data. In addition, CNN and RNN have been integrated for various application scenarios, such as image processing [2] and text classification [7]. However, none of them are designed for sequential recommendation.

In this paper, we propose a novel recurrent neural network model, namely **R**ecurrent **C**onvolutional **N**eural **N**etwork (RCNN), for sequential recommendation. It leverages the strengths of both the recurrent architecture of LSTM to capture complex long-term dependencies and the convolutional operation of CNN to extract local sequential patterns among hidden states. Specifically, we first generate a hidden state, *i.e*, a hidden sequential preference representation, at each time step by inputting a current item into the recurrent layer. Then, we treat recent hidden states of each time step as an "image" and search for local sequential features using horizontal and vertical convolutional filters. An intra-step horizontal filter is used to capture non-linear feature interactions, while an inter-step vertical filter is used for non-monotone local patterns. Moreover, the outputs of CNN and the hidden state vector of LSTM are concatenated together to describe user's overall interest, and then are fed into a fully-connected layer to generate a recommendation list. In summary, the main contributions of this paper are listed as follows:

- To the best of our knowledge, it is the first to propose to integrate both RNN and CNN to combine user's long- and short-term preference to generate a high-level hybrid representation for sequential recommendation.
- RCNN uses horizontal and vertical convolutional filters to learn high-order correlations among hidden state from locally to globally in a hierarchical way.
- We conduct experiments on three real datasets which demonstrate the superior performance of the proposed model with other state-of-the-art methods.

2 RELATED WORK

Different from general recommendation, sequential recommendation views the interactions as a sequence in time order and aims to predict the successive item(s) that a user is likely to interact with in the near future. Early sequential recommendation methods are based on Markov chains, which use sequential connections between users' actions to make prediction. A typical model FPMC [13] factorizes a transition matrix into item latent factors by integrating matrix factorization and Markov chains. HRM [18] further extends the idea by leveraging representations learning as latent factors. However, the above methods focus on modeling local sequential patterns between two adjacent actions while neglecting the global information of the whole sequences.

Recently, researchers have a surge of interest in applying deep learning to recommendation systems. Restricted Boltzmann Machine (RBM) [14] is one of the first neural networks applied to solve recommendation problems. Auto-encoders have also been a popular choice as the deep learning architecture of recommender systems [15, 24]. In addition, with the recent success of CNN on capturing local feature representation of images and texts [6, 7], it has shown a notable performance for top-n sequential recommendation. Caser [17] proposed to embed a sequence of recent items into "image" in the latent spaces and learn local features of the image as a sequential pattern using convolutional filters. However, CNN is capable of extracting local information while may fail to capture long-term dependencies.

On the other hand, as one of the most popular neural networks, Recurrent Neural Network (RNN) has been widely applied to model sequential data. Hidasi et al. applied RNN to model session data with a gated recurrent unit, which utilized session-parallel mini-batches and employed ranking loss functions for learning the model [4]. Tan et al. further proposed a data augmentation technique to improve the performance of RNN for session-based recommendation [16]. However, RNN holds a monotonicity assumption that the current item or hidden state is more important than the previous one, which impairs modeling of user's short-term interest. To deal with this problem, attentional mechanisms have been recently explored due to the ability of modeling user's attention. For example, NARM [8] was proposed as a hybrid encoder with an attention mechanism to emphasize user's main purpose in the current session. SHAN [22] was designed as a two-layer hierarchical attention network to take user-item and item-item interactions into account.

With the rapid development of deep neural networks, researchers further proposed to integrate CNN and RNN for various application scenarios. Lai et al. applied CNN to learn representation of texts and a bi-directional recurrent structure to capture contextual information for text classification [7]. Haque et al. designed a convolutional neural network as encoder and a multilayer LSTM network as decoder for image denoising [2]. However, none of these studies are designed for sequential recommendation and we propose to integrate RNN and CNN in an innovative way. The proposed RCNN utilizes the recurrent architecture of RNN to capture complex long-term dependencies, while employs the convolutional operation of CNN to extract short-term sequential patterns and item-specific features from the hidden states generated by RNN. With this integration method, RCNN is able to address sequential recommendation problems.

3 PROBLEM STATEMENT

In this section, we first introduce basic notations that will be used in this paper. Let $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ and $\mathcal{I} = \{i_1, i_2, ..., i_{|\mathcal{I}|}\}$ represent the sets of users and items respectively. Our task focuses on recommendation scenarios with implicit feedback [12, 13], which concerns whether a user $u \in \mathcal{U}$ is interacted with an item $i \in \mathcal{I}$ at time t. Each user is associated with a sequence of items from $\mathcal{I}$. By sorting interaction records in a chronological order, we can form the interaction sequence $\mathcal{I}^u = \{i_1^u, ..., i_t^u, ..., i_{|\mathcal{I}^u|}^u\}$ for user u, where $i_t^u \in \mathcal{I}$. The index t for i_t^u denotes the relative time index, instead of absolute time index in temporal recommendation like [13, 19].

With the above notations, we define the sequential recommendation task as follows. Given a user u's history sequence $\mathcal{I}^u$, we aim to infer the item(s) that user u would probably interact with in the future. Note that it is straightforward to convert the above task into *next-item* recommendation at time $t + 1$ or $next\text{-}future$ recommendation in the near future by replacing the next item with

the next item collection. For the sake of simplicity, we keep the *next-future* recommendation as the major task throughout the paper.

4 THE PROPOSED APPROACH

In this section, we introduce the proposed Recurrent Convolutional Neural Network model (RCNN), which incorporates a recurrent neural network to learn users' long-term interests and a convolutional neural network to learn users' sequential patterns and item-specific features among hidden states. As shown in Figure 1, RCNN consists of three components: a Recurrent Layer, Convolutional Layers, and a Fully-connected Layer. In the following subsections, we will introduce each part of our model and corresponding network learning strategy in detail.

4.1 Recurrent Layer

The first layer of RCNN is a long-short term memory network. LSTM [5], a variant of RNN, is able to alleviate the problem of "vanishing gradients" suffered from RNN by introducing a gate mechanism to hold long-term dependencies. Therefore, we use LSTM model in our approach for the concise and general purpose. Given the input $\mathbf{x}_t$ at the time t, the output $\mathbf{h}_t$ of the LSTM hidden layer is computed as follows.

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f),\\
\tilde{\mathbf{c}}_t &= tanh(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c),\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t,\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o),\\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t).
\end{aligned} \tag{1}$$

where $\mathbf{h}_0 = 0$, $\sigma(\cdot)$ is a logistic sigmoid function to do non-linear projection, and $\odot$ is the Hadamard product between two vectors. $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t$ and $\mathbf{c}_t$ are an input gate, a forget gate, an output gate and a cell state of the t-th object, respectively. $\mathbf{h}_t$ represents the hidden state vector to remember and store nodes of past states.

4.2 Convolutional Layers

The outputs of the recurrent layer are then fed simultaneously into a horizontal convolution layer and a vertical convolution layer. We present the previous k hidden states of user u at time t as a $k \times d$ matrix $\mathbf{E}^{(u,t)}$, where d is the hidden size and the rows of the matrix preserve the order of items. And then we regard the matrix $\mathbf{E}^{(u,t)}$ as an "image" to make the horizontal and the vertical convolution operations respectively. More details are explained below.

Horizontal Convolution on Hidden State. The convolution involves a filter vector sliding over a sequence, and picks up signals for sequential patterns at different time steps. The matrix $\mathbf{F} \in \mathbb{R}^{1\times w}$ is a filter for the horizontal convolution, where the height of a horizontal filter is 1 and the width equals to w. For each time step t in the sequence $\mathcal{I}^u$, we have a window matrix $\mathbf{E}^{(u,t)} \in \mathbb{R}^{1\times d}$ with a single hidden state (*i.e.*, $\mathbf{E}^{(u,t)} = h_t$). The result of the interaction for window matrix $\mathbf{E}^{(u,t)}$ is produced as follows:

$$v_t = \sum_{j=0}^{d-w+1} f(\mathbf{E}^{(u,t)}_{j:j+w-1} \cdot \mathbf{F} + b) \tag{2}$$

where $\cdot$ is the inner product, j is the position of $\mathbf{E}^{(u,t)}$'s row vector, $b \in \mathbb{R}$ is a bias term and f is the ReLU activation function. Moreover, we use n horizontal filters with the same size to generate multiple convolution value. The final horizontal convolution result $\mathbf{o}_t \in \mathbb{R}^n$ is

$$\mathbf{o}_t = [v_t^1\ v_t^2\ ...\ v_t^n] \tag{3}$$

where v_t^l is the convolution value generated with the l-th filter and $\mathbf{o}_t$ is a new feature representation generated from n filters for the window vector at time step t and is considered as the current item's feature interaction.

Vertical Convolution on Hidden State. We use tilde(~) for the symbols of the vertical convolution. We denote the vertical filter $\tilde{\mathbf{F}} \in \mathbb{R}^{\tilde{k}\times 1}$ for the operation of vertical convolution. For each time step t in the sequence $\mathcal{I}^u$, we have a window matrix $\tilde{\mathbf{E}}^{(u,t)} \in \mathbb{R}^{\tilde{k}\times d}$ with $\tilde{k}$ consecutive items, denoted as:

$$\tilde{\mathbf{E}}^{(u,t)} = [\mathbf{h}_{t-\tilde{k}+1}, ..., \mathbf{h}_{t-1}, \mathbf{h}_t] \tag{4}$$

where each commas represents a row vector concatenation and the full width of $\tilde{\mathbf{E}}$ equals to d. $\tilde{\mathbf{F}}$ interacts with each column of $\tilde{\mathbf{E}}^{(u,t)}$ by sliding d times from left to right for user u at time step t in a valid way, and finally yielding the vertical convolution result $\tilde{\mathbf{v}}_t \in \mathbb{R}^d$.

$$\tilde{\mathbf{v}}_t = [\tilde{v}_1\ \tilde{v}_2\ ...\ \tilde{v}_d] \tag{5}$$

$$\tilde{\mathbf{o}}_t = \tilde{\mathbf{v}}_t \odot \mathbf{h}_t. \tag{6}$$

where $\tilde{\mathbf{v}}_t$ is equal to the weights of recent items' hidden features and $\tilde{\mathbf{o}}_t$ is the final output of the vertical convolution, which denotes user u's local sequential pattern at time t. With one vertical filter we can learn to enhance the impact of the k previous items' hidden states on current interest for user u, which highlights user's short-term interest within recent $\tilde{k}$ time steps.

4.3 Fully-connected Layer

To capture the current overall preference of user u, we concatenate the outputs of the two convolutional layers and the hidden vector of LSTM together, and project them to a fully-connected layer with $\mathcal{I}$ nodes, written as

$$\mathbf{y}^{(u,t)} = \sigma(\mathbf{W}' \begin{bmatrix} \mathbf{o}_t \\ \mathbf{h}_t \\ \tilde{\mathbf{o}}_t \end{bmatrix} + \mathbf{b}') \tag{7}$$

where $\mathbf{W}' \in \mathbb{R}^{|\mathcal{I}|\times(2d+n)}$ and $\mathbf{b}' \in \mathbb{R}^{|\mathcal{I}|}$ are the weight matrix and the bias term for the output layer, respectively. $\mathbf{h}_t$ intends to capture user's long-term preferences, whereas $\mathbf{o}_t$ learns item-specific features and $\tilde{\mathbf{o}}_t$ captures short-term sequential patterns. The value $y_i^{(u,t)}$ is associated with the probability of how likely user u will interact with item i at time step t.

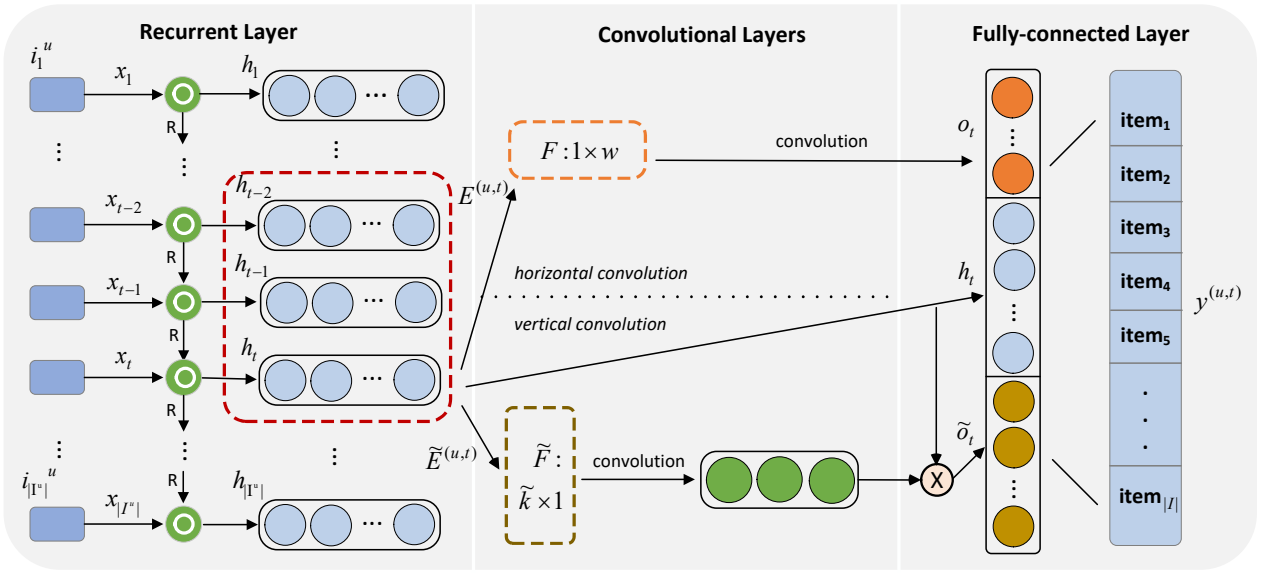


Figure 1: The overall architecture of RCNN. The red dash rectangular box represents the window matrix, while the orange and brown dash rectangular boxes are horizontal and vertical convolutional filters.

4.4 Network Learning

At last, we adapt cross-entropy as our loss function for model optimization. The objective function to be minimized in the model optimization is define as follows:

$$\mathcal{L} = -\sum_{u}^{|\mathcal{U}|}\sum_{t}^{|\mathcal{I}^u|} \hat{\mathbf{y}}^{(u,t)} log(\mathbf{y}^{(u,t)}) + (1-\hat{\mathbf{y}}^{(u,t)}) log(1-\mathbf{y}^{(u,t)}) + \lambda||\Theta||^2 \tag{8}$$

where $\mathbf{y}^{(u,t)}$ is the probability of all items interacted by user u at time $t+1$, and $\hat{\mathbf{y}}^{(u,t)}$ denotes a one-hot vector. If item i is interacted at time $t+1$, $\hat{\mathbf{y}}_i^{(u,t)} = 1$. Otherwise, $\hat{\mathbf{y}}_i^{(u,t)} = 0$. Θ represent all of the model parameters that are be learned and λ is the regularization weight .

5 EXPERIMENTS

In this section, we first introduce the experiment settings. And then we conduct experiments to answer the following research questions:

Q1: What is the performance of our model RCNN, comparing to other state-of-the-art methods?

Q2: How much do the horizontal and vertical convolutional operations help RCNN?

Q3: How do the parameters, such as the size of convolutional filters, affect the performance of RCNN?

5.1 Experimental Setup

5.1.1 Datasets. We study the effectiveness of our proposed approach on three public datasets with different kinds of items. Gowalla[1] and Foursquare[2] are two widely used check-ins datasets, which contains user-POI check-ins information in the location-based social networking sites (LBSN). Taobao is a user-purchase dataset obtained from IJCAI 2017 competition [3]. We eliminated users interacting with fewer than 15 items and items interacted by fewer than 10 users in the three datasets. After preprocessing, the basic statistics of all datasets are listed in Table 1.

Table 1: Statistics of the three datasets.

Dataset	#users	#items	#feedbacks	avg. seq. len	sparsity(%)
Gowalla	5,073	7,020	252,944	49.86	99.29
Foursquare	2,031	3,113	106,229	52.30	98.32
Taobao	2,481	1,419	72,553	29.24	97.94

5.1.2 Evaluation Metrics. For the two recommendation tasks, i.e., next-item and next-future recommendation, we divided the datasets into training/validation/test set with a 98/1/1 split and 70/10/20 split, respectively. In addition, we remove duplicate items from the test set.

To evaluate the performance of each approach, we adopt a variety of metrics widely used in previous works [4, 13], including *Hit Ratio*(HR@N), *Mean Average Precision* (MAP), *Precision*@N, and *Recall*@N. In our experiments, we choose $N = \{1, 5, 10\}$. To make results more stable, we repeated each experiment five times and used the average value of each metrics.

5.1.3 Baselines. We consider the following baselines for performance comparisons:

[1] http://snap.stanford.edu/data/loc-gowalla.html

[2] https://sites.google.com/site/yangdingqi

[3] https://tianchi.shuju.aliyun.com/datalab/dataSet

- **BPR-MF** [12] is a state-of-the-art method for non-sequential recommendation, which optimizes Matrix Factorization model using a pairwise ranking loss.
- **FPMC**[4][13] is a classic hybrid model combing MC and MF for next-basket recommendation. FPMC allows a basket of several items at each step. For our sequential recommendation, each basket has a single item.
- **Caser**[5] [17] is the first method of leveraging convolutional filters to learn users' sequential patterns for sequential recommendation. It incorporates the convolutional neural network with a latent factor model.
- **RUM**[6] [20] utilizes external memories to improve sequential recommendation, which contains two variants, item-level (RUM^I) and feature-level (RUM^F).
- **GRU** [1] is an extension of RNN [23], which is equipped with two gates to control the information flow for sequential recommendation.
- **LSTM** [5] is another variant of RNN, which contains a memory cell and three multiplicative gates to hold long-term dependencies.
- **NARM**[7] [8] is an RNN-based state-of-the-art model, which employs an item-level attention mechanism to capture user's main purpose from hidden states and combines it with the sequential behavior as user's final representation to generate recommendation.

5.2 Comparison of Performance

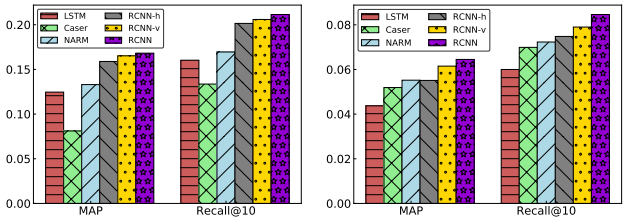
The results of different methods for both recommender tasks are illustrated in Table 2. It can be observed that:

First, among all baseline models, BPR-MF has the most unfavorable performance in nearly all the cases, since BPR-MF is an non-sequential recommendation method. The classic model FPMC performs better than BPR-MF on most cases by taking the sequential correlation between adjacent actions into account. The experimental results verify that sequential information can help improve the recommendation performance in real-world systems.

Second, both RNNs-based methods (*i.e.*, GRU, LSTM and NARM) and memory-based models (*i.e.*, Caser and RUM) improve the effectiveness by using deep neural networks. This shows the ability of neural networks on modeling users' general tastes and their sequential behaviors. On Foursquare and Gowalla datasets, both Caser and RUM achieve better performances than LSTM and GRU. But they get much worse performances on Taobao dataset. In fact, Taobao dataset have more sequential signals than Foursquare and Gowalla datasets. Therefore, both LSTM and GRU can achieve favorable performances, because of their advantages in modeling long-term dependencies. It calls us to combine a deep model with recurrent structure for improvements.

Third, in terms of memory-based methods, RUM performs better than Caser under the metric of Prec@N on Taobao dataset, yet performs worse under the metrics of Recall@N and MAP on most datasets. Overall, RUM and Caser perform well, which indicates that both KV-MN architecture and convolutional operations can capture users' preferences. Furthermore, the state-of-the-art attention-based RNN model NARM beats all the other baselines on all datasets owing to the effectiveness of the attentive mechanism. This shows the effectiveness of the attentive mechanism to capture user's evolving appetite for items.

Finally, we compare our proposed model RCNN with all the baselines. It is clear to see that RCNN consistently and significantly outperforms these baselines on all datasets. This shows that leveraging the power of convolutional neural network, RCNN can model users' sequential pattern better and capture many important features, resulting in a better recommendation. Moreover, we also observe that the performances of most models degrade as the datasets become sparser.



(a) Taobao (b) Gowalla

Figure 2: Performance with variants of RCNN in terms of Rec@10 and MAP on two datasets.

5.3 Influence of Components

To evaluate the contribution of convolutional filters to form user's final hybrid representation, we conduct experiments to analyze each component of convolution. Figure 2 shows MAP and Recall@10 on Taobao and Gowalla. (We obtained the similar results on another dataset.) RCNN-h and RCNN-v denote a downgrade version of RCNN that only uses horizontal and vertical convolution, respectively. From Figure 2, we can observe that Caser outperforms LSTM on Gowalla, while LSTM performs better than Caser on Taobao. This suggests that Caser's performance is greatly affected by the sequential signals of the data. NARM performs better than Caser and LSTM on both datasets due to the attention mechanism. However, NARM performs worse than RCNN-v, which indicates that attention mechanism of vertical convolution plays a better impact than NARM due to use of the filter and nonlinear function. RCNN-h performs better than NARM on most cases, suggesting the feature interactions of items are also important for sequential recommendation. Overall, our final model RCNN performs the best on the two datasets.

5.4 Influence of Hyper-parameters

In this subsection, we investigate the impact of the parameters w and $\tilde{k}$ in our model. Due to space limit, we only show MAP on Foursquare and Taobao datasets. We apply grid search over combinations of $w \in [8, 16, 32, 64, 128]$ and $\tilde{k} \in [2, 3, 4, 5, 6]$ for a better recommendation performance. The experiment results are shown

[4]https://github.com/khesui/FPMC
[5]https://github.com/graytowne/caser_pytorch
[6]https://github.com/ch-xu/RUM
[7]https://github.com/lijingsdu/sessionRec_NARM

Table 2: Performance comparison with baselines on all three datasets for next-item and next-future recommendation. The best performance is highlighted in boldface (higher is better).

Datasets	Methods	Next-item Recommendation				Next-future Recommendation						
		HR@1	HR@5	HR@10	MAP	Prec@1	Prec@5	Prec@10	Recall@1	Recall@5	Recall@10	MAP
Gowalla	BPR-MF	0.0185	0.0481	0.0645	0.0382	0.1426	0.0787	0.0539	0.0128	0.0330	0.0450	0.0315
	FPMC	0.0219	0.0582	0.0755	0.0421	0.1451	0.0788	0.0563	0.0146	0.0342	0.0450	0.0349
	Caser	0.0231	0.0674	0.0954	0.0486	0.1579	0.0906	0.0672	0.0164	0.0455	0.0699	0.0519
	RUM^I	0.0351	0.0684	0.0863	0.0537	0.1547	0.0846	0.0597	0.0139	0.0380	0.0512	0.0380
	RUM^F	0.0337	0.0670	0.0887	0.0543	0.1570	0.0799	0.0565	0.0149	0.0347	0.0480	0.0360
	GRU	0.0394	0.0997	0.1415	0.0738	0.1401	0.0858	0.0643	0.0133	0.0404	0.0594	0.0429
	LSTM	0.0428	0.1080	0.1380	0.0773	0.1491	0.0884	0.0652	0.0139	0.0405	0.0600	0.0437
	NARM	0.0559	0.1307	0.1715	0.0995	0.1892	0.1034	0.0765	0.0189	0.0506	0.0730	0.0552
	RCNN	**0.0574**	**0.1427**	**0.1950**	**0.1031**	**0.2107**	**0.1290**	**0.0961**	**0.0211**	**0.0579**	**0.0846**	**0.0645**
	Improv.	2.68%	9.18%	13.70%	3.62%	11.36%	24.76%	25.62%	11.64%	14.43%	15.89%	16.85%
Foursquare	BPR-MF	0.0389	0.0926	0.1216	0.0677	0.1201	0.0754	0.0796	0.0095	0.0209	0.0413	0.0268
	FPMC	0.0399	0.0822	0.0955	0.0621	0.1952	0.1013	0.0621	0.0161	0.0393	0.0476	0.0371
	Caser	0.0177	0.0448	0.0635	0.0356	0.1674	0.0951	0.0721	0.0179	0.0429	0.0602	0.0482
	RUM^I	0.0592	0.1172	0.1418	0.0902	0.2068	0.1122	0.0809	0.0163	0.0402	0.0572	0.0451
	RUM^F	0.0605	0.1236	0.1472	0.0957	0.2319	0.1120	0.0773	0.0185	0.0423	0.0558	0.0460
	GRU	0.0443	0.0980	0.1280	0.0738	0.1327	0.0775	0.0584	0.0101	0.0301	0.0448	0.0343
	LSTM	0.0409	0.1024	0.1305	0.0732	0.1574	0.0930	0.0683	0.0124	0.0344	0.0502	0.0399
	NARM	0.0611	0.1364	0.1713	0.1006	0.2462	0.1268	0.0914	0.0191	0.0484	0.0683	0.0547
	RCNN	**0.0694**	**0.1487**	**0.1984**	**0.1124**	**0.2748**	**0.1477**	**0.1040**	**0.0205**	**0.0537**	**0.0744**	**0.0602**
	Improv.	13.58%	9.02%	15.82%	11.73%	11.62%	16.39%	13.79%	7.33%	10.95%	8.93%	10.05%
Taobao	BPR-MF	0.0508	0.1749	0.2656	0.1176	0.1608	0.1309	0.1052	0.0170	0.0735	0.1208	0.0807
	FPMC	0.0601	0.1809	0.2447	0.1225	0.2126	0.1416	0.1020	0.0235	0.0764	0.1139	0.0808
	Caser	0.0290	0.0842	0.1189	0.0581	0.1374	0.0888	0.0669	0.0277	0.0885	0.1336	0.0811
	RUM^I	0.0705	0.2209	0.3249	0.1530	0.2112	0.1382	0.1083	0.0248	0.0786	0.1238	0.0842
	RUM^F	0.0826	0.2390	0.3370	0.1649	0.2140	0.1487	0.1125	0.0245	0.0849	0.1288	0.0866
	GRU	0.1511	0.3108	0.3906	0.2321	0.4306	0.2063	0.1357	0.0465	0.1148	0.1544	0.1195
	LSTM	0.1737	0.3527	0.4208	0.2593	0.4346	0.2123	0.1405	0.0491	0.1191	0.1603	0.1246
	NARM	0.1943	0.3724	0.4478	0.2834	0.4486	0.2195	0.1489	0.0492	0.1233	0.1695	0.1330
	RCNN	**0.2656**	**0.4752**	**0.5506**	**0.3672**	**0.5566**	**0.2804**	**0.1832**	**0.0622**	**0.1589**	**0.2114**	**0.1682**
	Improv.	36.70%	27.60%	22.96%	29.57%	24.07%	27.74%	23.04%	26.42%	28.87%	24.72%	26.47%

Table 3: The performance of RCNN with varying w and $\tilde{k}$ in terms of MAP on Foursquare and Taobao.

Dataset	MAP: w \ $\tilde{k}$	$\tilde{k}=2$	$\tilde{k}=3$	$\tilde{k}=4$	$\tilde{k}=5$	$\tilde{k}=6$
Foursquare	w =8	0.0577	0.0566	0.0587	0.0566	0.0564
	w =16	0.0567	0.0589	0.0577	0.0580	0.0570
	w =32	0.0584	0.0577	0.0594	0.0575	0.0569
	w =64	0.0598	0.0590	**0.0602**	0.0579	0.0574
	w =128	0.0596	0.0580	0.0578	0.0574	0.0574
Taobao	w =8	0.1671	0.1673	0.1668	0.1665	0.1671
	w =16	0.1672	0.1675	0.1687	0.1692	0.1672
	w =32	0.1675	0.1674	0.1696	0.1685	0.1686
	w =64	0.1671	0.1693	0.1693	0.1684	0.1682
	w =128	**0.1698**	0.1695	0.1678	0.1672	0.1695

in Table 3. We can see that either fixing w or fixing $\tilde{k}$, the performance of our model is improved first and then deteriorated on most cases. This suggests that a proper size of filters can capture users' short-term interest and high-order feature interactions better. Consequently, the best sizes are chosen as $w = 64, \tilde{k} = 4$ on Foursquare and $w = 128, \tilde{k} = 2$ on Taobao. $\tilde{k}$ on Foursquare is larger than the value on Taobao. It may be because the average sequence length on Foursquare is longer.

6 CONCLUSIONS

In this paper, we proposed a novel **R**ecurrent **C**onvolutional **N**eural **N**etwork model (RCNN), which leverages the strengths of both RNN and CNN for sequential recommendation. RCNN utilizes the recurrent architecture of RNN to capture complex long-term dependencies, while uses the convolutional operation of CNN to extract short-term sequential patterns. Our experimental results on three real-world datasets show that RCNN significantly outperformed the state-of-the-art approaches on sequential recommendation.

7 ACKNOWLEDGMENTS

This research was partially supported by the NSFC (61876117, 61876217, 61872258, 61728205), the Suzhou Science and Technology Development Program (SYG201803) and the Open Program of Neusoft Corporation (SKLSAOP1801).

REFERENCES

[1] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[2] Kazi Nazmul Haque, Mohammad Abu Yousuf, and Rajib Rana. 2018. Image denoising and restoration with CNN-LSTM Encoder Decoder with Direct Attention. *arXiv preprint arXiv:1801.05141* (2018).
[3] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *SIGIR*. ACM, 355–364.
[4] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).
[5] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[7] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification.. In *AAAI*, Vol. 333. 2267–2273.
[8] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. ACM, 1419–1428.
[9] Qiang Liu, Shu Wu, and Liang Wang. 2017. Multi-behavioral sequential prediction with recurrent log-bilinear model. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1254–1267.
[10] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent

neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 130–137.
[11] S. Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.
[12] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.
[13] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. ACM, 811–820.
[14] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*. 791–798.
[15] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *WWW*. 111–112.
[16] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 17–22.
[17] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. ACM, 565–573.
[18] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for NextBasket Recommendation. In *SIGIR*. 403–412.
[19] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. ACM, 495–503.
[20] Chen X, Xu H, Zhang Y, Tang J, Cao Y, Qin Z, and Zha H. 2018. Sequential recommendation with user memory networks. In *WSDM*. ACM, 108–116.
[21] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*. 3119–3125.
[22] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential Recommender System based on Hierarchical Attention Networks. In *IJCAI*. 3926–3932.
[23] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. ACM, 729–732.
[24] Alice X. Zheng, Alice X. Zheng, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *WSDM*. 153–162.