

# Rethinking Convolutional Networks for Attribute-Aware Sequential Recommendation

Shereen Elsayed<sup>1,2</sup>, Ngoc Son Le<sup>1,2</sup>, Ahmed Rashed<sup>3</sup> and Lars Schmidt-Thieme<sup>1,2</sup>

<sup>1</sup>Information Systems and Machine Learning Lab (ISMILL), University of Hildesheim, Germany

<sup>2</sup>VWFS Data Analytics Research Center (VWFS DARC), University of Hildesheim, Germany

<sup>3</sup>Volkswagen Financial Services AG, Germany

{elsayed,sle,schmidt-thieme}@ismill.de, ahmed.galal.ahmed.rashed@vwfs.com

## Abstract

Attribute-aware sequential recommendation entails predicting the next item a user will interact with based on a chronologically ordered history of past interactions, enriched with item attributes. Existing methods typically leverage self-attention mechanisms to aggregate the entire sequence into a unified representation used for next-item prediction. While effective, these models often suffer from high computational complexity and memory consumption, limiting their ability to process long user histories. This constraint restricts the model's capacity to fully capture long-term user preferences. In some scenarios, modeling item interactions purely through attention may also not be the most effective approach to extract sequential patterns. In this work, we propose **ConvRec**, an alternative method with linear computational and memory complexity that employs convolutional layers in a hierarchical, down-scaled fashion to generate compact, yet expressive sequence representations. To further enhance the model's ability to capture diverse sequential patterns, each layer aggregates the neighboring items gradually to reach a comprehensive sequence representation. Extensive experiments on four real-world datasets demonstrate that our approach outperforms state-of-the-art sequential recommendation models, highlighting the potential of convolution-based architectures for efficient and effective sequence modeling in recommendation systems. Our implementation code and datasets are available at <https://github.com/ismill-research/ConvRec>.

## 1 Introduction

In the era of explosive growth of social media networks, video streaming, and e-commerce platforms, recommendation systems have become an indispensable tool to help users navigate the vast digital landscape by providing personalized suggestions that align with their unique prefer-

<sup>†</sup>The extended version of this paper, including supplementary appendices, is available at <https://arxiv.org/pdf/2605.04723>.

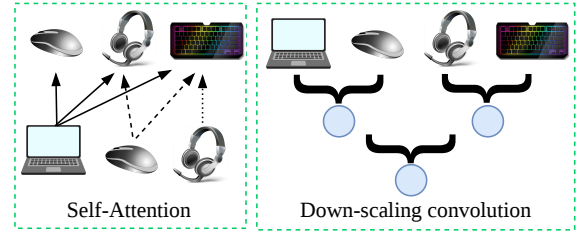


Figure 1: Illustration of the self-attention versus the down-scaling convolution mechanism.

ences and past behaviors. More recently, sequential recommendation has emerged as a dominant recommendation task. Unlike traditional collaborative filtering (CF) methods, such as Matrix Factorization (MF) [Koren *et al.*, 2009; Rendle *et al.*, 2012] that solely focus on static user-item interactions, sequential recommendation models are designed to capture the temporal dynamics of user behaviors.

Earlier approaches in sequential recommendation domain were based on Markov Chains [Shani *et al.*, 2005], [Rendle *et al.*, 2010], which could capture short-term dependencies but struggled with longer-range patterns. With the arrival of deep learning, a new wave of model architectures that are capable of handling high-order, nonlinear, and long-range data sequences was introduced. Models using Recurrent Neural Networks (RNNs) [Cho *et al.*, 2014], [Hidasi and Karatzoglou, 2018] could effectively capture longer evolving sequential dependencies, but suffer from training and inference inefficiency due to the lack of parallelization. On the other hand, Transformer-based models managed to address these limitations by leveraging self-attention to model dependencies across all items in a sequence at the same time, which eliminates the need for sequential processing and enables full parallelization during training. Models, such as SASRec [Kang and McAuley, 2018], BERT4Rec [Sun *et al.*, 2019], helped establish self-attention as a dominant modeling paradigm in the field of sequential recommendation. Despite parallelization capabilities, the computational and memory complexity of self-attention scales quadratically with the sequence length, which offsets much of the training efficiency it provides.

In this paper, we revisit Convolutional Neural Networks (CNNs) as a competitive framework for temporal and attribute-aware sequential recommendation. Although early CNN-based models like Caser [Tang and Wang, 2018] demonstrated the ability to capture local transition patterns, its receptive field is strictly bounded within the fixed history window of the last few items. Thus, it is unable to capture the global view of the long-range sequential dependencies in the way Transformers can. To overcome this limitation, we propose a hierarchical convolutional encoder that progressively downsamples the entire interaction sequence (via striding/pooling) to a single node, thereby distilling the temporal features into a compact “intent” representation (Figure 1) with a global view of the sequence. This design takes inspiration from the pyramidal and patch-based techniques that were successfully used in time-series and vision models [Wang *et al.*, 2024], [Lin *et al.*, 2017].

We introduce **ConvRec**, a hierarchical down-scaling convolutional model for attribute-aware sequential recommendation. First, it fuses each item embedding in the user’s interaction history with its corresponding attribute embeddings and contextual embeddings into a joint representation. These embeddings form a chronological interaction sequence for each user, which is fed through a series of convolutional blocks. In each block, a strided convolution operation aggregates local neighboring interactions and downsamples the sequence. By repeating this down-scaling procedure in a pyramidal fashion, ConvRec progressively distills the entire user interaction history into a single vector with rich representation of the entire sequence, which is then used to score and rank next-item candidates via a dot-product operation.

Our primary contributions are as follows:

- We propose a novel convolutional encoder for sequential recommendation, which leverages item attributes and context, and applying a hierarchical down-scaling convolutions to capture both local and long-range user interaction patterns.
- Experiments across widely used sequential recommendation datasets, demonstrating superior performance of ConvRec in comparison to state-of-the-art baselines.
- We show that ConvRec scales linearly in memory and computation time with sequence length, and offers substantial efficiency improvements over existing best performing Transformer-based models on long user sequences.

## 2 Related Work

**Sequential Recommendation.** Sequential recommendation aims to predict the next item a user will interact with and is central to modern recommender systems. Earlier models like Caser [Tang and Wang, 2018], CosRec [Yan *et al.*, 2019], and RCNN [Xu *et al.*, 2019] used convolutions and recurrent layers to capture sequential patterns. GRU4Rec [Jannach and Ludewig, 2017] and BERT4Rec [Sun *et al.*, 2019] introduced recurrent and transformer architectures, while SASRec [Kang and McAuley, 2018] employed multi-head self-attention. Extensions such as TiSASRec [Li *et al.*, 2020], CoSeRec [Liu

*et al.*, 2021b], and TiCoSeRec [Dang *et al.*, 2023] incorporate temporal and contrastive learning signals to further improve performance. **Attribute-aware Sequential Recommendation.** Incorporating item attributes is essential for capturing user preferences more accurately.  $S^3$ Rec [Zhou *et al.*, 2020] leverages pre-training to enhance data representations. Other models, such as NOVA [Liu *et al.*, 2021a], DIFSR [Xie *et al.*, 2022], and MSSR [Lin *et al.*, 2024], propose various strategies for feature fusion. CARCA [Rashed *et al.*, 2022] introduces cross-attention mechanisms to compute item relevance scores more effectively. ProxyRCA [Seol *et al.*, 2024] builds upon CARCA by incorporating proxy-based item embeddings, significantly improving representation quality. Additionally, HGTL [Xu *et al.*, 2025] applies cross-domain transfer learning to leverage user preferences from multiple domains.

Despite these advancements, convolutional architectures remain relatively underexplored in the context of sequential recommendation, even though they have shown strong potential for capturing local patterns in sequences [Liu *et al.*, 2022; Wang *et al.*, 2024]. In this work, we propose a lightweight, hierarchical convolutional approach for down-scaling and encoding user sequences, offering an efficient and effective alternative for sequence representation.

## 3 Problem Setting

In the attribute-aware sequential recommendation setting, we consider a set of users  $\mathcal{U}$  interacting with a set of items  $\mathcal{I}$ . Each user  $u \in \mathcal{U}$  has a chronologically ordered interaction sequence  $S^u = [i_1^u, \dots, i_{|S^u|}^u]$ , where each item is associated with a set of descriptive features/attributes  $A$  (e.g., title, brand, etc.), the items features sequence for user  $u$  is denoted as  $[a_1^u, \dots, a_{|S^u|}^u]$ . Additionally, each interaction is accompanied by contextual information  $C$ , such as the timestamp of the interaction, which can be decomposed into temporal components (e.g., year, month, day). We denote the contextual sequence for user  $u$  as  $[c_1^u, \dots, c_{|S^u|}^u]$ . The primary objective of the next-item recommendation task is to predict the most likely item a user will interact with at the next time step.

## 4 ConvRec

Considering the similarity between time-series forecasting and next-item recommendation tasks, sequential recommendation can similarly benefit from the hierarchical scaling properties used in time series. Specifically, a user’s input sequence can be aggregated in a hierarchically down-scaled manner to derive a meaningful sequence representation. Given that neighboring items in a sequence are often more closely related than distant ones [Wang *et al.*, 2020; Kang and McAuley, 2018], unlike the self-attention mechanism, we propose a representation method where items are gradually merged with their neighbors to form a unified representation of the entire sequence.

### 4.1 Item Encoding

For item encoding, we rely not only on item IDs, but also emphasize the importance of incorporating item attributes/features. These attributes provide insight into user

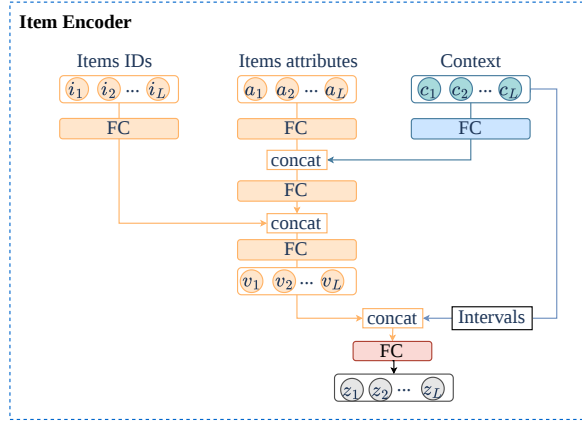


Figure 2: Item encoding module.

preferences and enhance the model’s ability to accurately predict the next item. For a given input sequence of fixed length  $L$  (obtained through either padding or truncation), each item’s features are passed through a fully connected layer:

$$a_i'^u = a_i^u W_a + b_a \quad (1)$$

$W_a \in \mathbb{R}^{|A| \times d_a}$  is a weight matrix,  $|A|$  is the length of the item’s features set,  $d_a$  is the embedding dimension of the input features and  $b_a$  is the bias term. In addition to item features, we incorporate interaction context, such as time and location of interactions. In this work, we focus on temporal context, which is also encoded via a fully connected layer:

$$c_i'^u = c_i^u W_c + b_c \quad (2)$$

$W_c \in \mathbb{R}^{|C| \times d_c}$  is the weight matrix for the context features,  $|C|$  is the number of contextual elements,  $d_c$  is the context embedding dimension, and  $b_c$  is the bias term. The encoded item features and context vectors are then concatenated and passed through another fully connected layer to generate the interaction embedding:

$$f_i^u = \text{Concat}(a_i'^u, c_i'^u)W_f + b_f \quad (3)$$

Here,  $W_f \in \mathbb{R}^{(d_a+d_c) \times d_f}$ ,  $d_f$  is the output embedding dimension, and  $b_f$  is the bias term.

While various strategies exist for incorporating item IDs in recommendation systems, we adopt the approach introduced in [Seol *et al.*, 2024], which assigns unique embeddings only to the most frequent items. Less frequent items are assigned a shared generic ID. This approach shifts the model’s learning focus toward high-frequency items while relying on feature-based representations for infrequent ones. Accordingly, we embed item IDs using a lookup table:  $i_i^{\prime u} = \text{LookupTable}(i_i^u)$ . This embedding is concatenated with the previously obtained feature-context embedding and passed through a linear layer:

$$v_i^u = \text{Concat}(i_i'^u, f_i^u)W_v + b_v \quad (4)$$

$W_v \in \mathbb{R}^{(d_i+d_f) \times d_v}$ ,  $d_v$  is the embedding dimension of the combined item IDs and side information, and  $b_v$  is the bias term.

In sequential recommendation, temporal dynamics play a critical role. The time gap between interactions can vary significantly, so time not only contributes to the representation of individual interactions but can also help assess their relative importance. To this end, we compute time intervals (year, month, day) between each two consecutive interactions, and concatenate the intervals to the input sequence, where for the first interaction time interval is set to zero:

$$\Delta \mathbf{c}^u = \text{Concat}(\Delta \mathbf{c}^0, [\mathbf{c}_{2:L}^u - \mathbf{c}_{1:L-1}^u]), \quad (5)$$

$$Q^u = \text{LayerNorm}(\sigma(\text{Concat}(V^u, \Delta \mathbf{c}^u))) \quad (6)$$

Here,  $c$  is the interaction context where we extract only the **(year, month and day)** for the interval calculation,  $V^u = [v_1, v_2, \dots, v_L]$  is the sequence of item representations, and  $\Delta \mathbf{c}^u = [\Delta c_1^u, \Delta c_2^u, \dots, \Delta c_L^u]$  are the corresponding intervals, and  $\sigma$  denotes the GELU activation function [Hendrycks and Gimpel, 2016]. Then it is fed into a fully connected layer to obtain a combined representation.

$$Z^u = \sigma(Q^u W_Q + b_Q) \quad (7)$$

$W_Q \in \mathbb{R}^{(d_v+3) \times d_v}$  is a weight matrix, and  $b_Q$  is the bias term, and  $\sigma$  denotes the GELU activation function.

## 4.2 Convolution Down-scaling Component (CDS)

Building on the idea of gradual aggregation within a user’s historical interaction sequence, we design a hierarchy of convolutional layers to construct the final sequence representation. Given an encoded sequence  $Z^u \in \mathbb{R}^{L \times d_v}$ , where  $L$  denotes the maximum sequence length and  $d_v$  is the dimensionality of the interaction embeddings, our objective is to reduce the sequence length  $L$  to 1. This is achieved by progressively aggregating information through a series of hierarchical convolutional downscaling layers that downscale the sequence over multiple ConvBlocks:

$$\begin{aligned} X^u &= \mathbf{CDS}(Z^u) \\ &= \text{ConvBlock}_M \circ \dots \circ \text{ConvBlock}_1(Z^u) \end{aligned} \quad (8)$$

Here,  $Z^u$  denotes the input embeddings, initially shaped as  $L \times d_v$ , where  $L$  is the sequence length and  $d_v$  is the embedding dimension, and  $X^u$  denotes the output representation. To enable convolutional operations across neighboring items, the embeddings are reshaped such that the convolutional kernel can operate along the sequence dimension with  $d_v$  input channels.

**Convolution Block (ConvBlock).** To perform the convolution operations effectively, the input sequence  $Z^u$  is transposed to shape  $d_v \times L$ , treating  $d_v$  as the number of input channels. This reshaping allows 1D convolution to be applied along the sequence dimension  $L$ :

$$K^{(j)u} = \text{1D-Conv}(\text{Padding}(O^{(j-1)uT}), \text{KernelSize}, \text{Stride}),$$

**where**  $O^{(0)u} = Z^u$

$$(9)$$

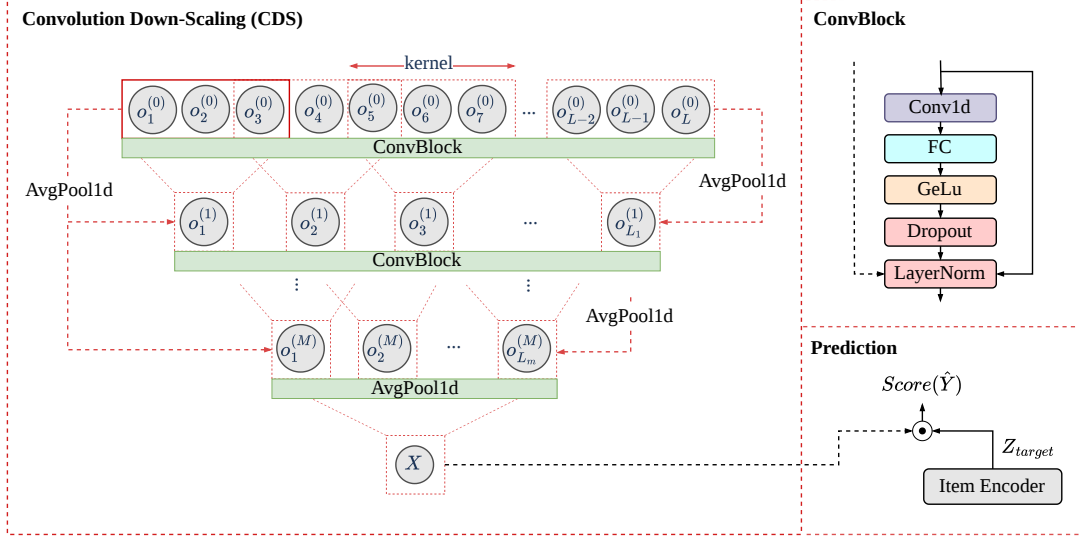


Figure 3: The ConvRec framework.

The 1D convolution is applied across the sequence dimension ( $L$ ) using a specified kernel size and stride. In most cases, the stride is set equal to the kernel size to ensure non-overlapping windows, which empirically yielded better performance. When the sequence length is not divisible by the kernel size, appropriate padding is applied to maintain dimensional consistency. After convolution,  $K^{(j)u}$  is transposed back to the shape  $L^{(j)} \times d_v$ , where  $L^{(j)}$  is the sequence length after applying the convolution operation. A final fully connected layer is then applied:

$$G^{(j)u} = \sigma \left( K^{(j)uT} W_G + b_G \right) \quad (10)$$

Here,  $W_G \in \mathbb{R}^{d_{K(j)} \times d_{G(j)}}$  is the weight matrix of the output projection layer,  $b_G$  is the bias term, and  $d_G$  is the output embedding dimension, which is also typically set equal to  $d_v$  for consistency and  $\sigma$  is a GELU activation.

Subsequently, layer normalization is performed on the output. Additionally, we incorporate two types of weighted residual connections to enhance training stability:

$$O^{(j)u} = \text{LayerNorm} \left( G^{(j)u} + \alpha_1 \cdot \text{AvgPool}(Z^u) + \alpha_2 \cdot \text{ProgRes}^{(j-1)} \right) \quad (11)$$

The first residual connection is derived by applying average pooling to the original input. This helps preserve the original signal and mitigates the risk of vanishing gradients. The second,  $\text{ProgRes}^{(j)}$ , is a progressive residual connection from the previous convolutional layer's output. It is also downsampled using average pooling to match the output dimensions.

In the first convolutional layer, both residual connections originate from the same input, as no prior layer exists. However, in subsequent layers,  $\text{ProgRes}$  continues to evolve, al-

lowing for smoother gradient flow and improved model convergence.  $\alpha_1$  and  $\alpha_2$  are learnable parameters that are initialized to 0.5.

Consequently, the downscaling convolutional blocks can be defined as follows:

**for**  $j = 1, \dots, M$  **do**:

$$O^{(j)u} = \text{ConvBlock}_j(\text{Padding}(O^{(j-1)u}), \text{KernelSize}, \text{Stride}),$$

**where**  $O^{(0)u} = Z^u$ .

(12)

Here,  $M$  denotes the total number of convolutional layers, resulting in the final output  $O^{(M)u}$  with an expected shape of  $d_v \times 1$ . However, in cases where the convolutional layers do not completely collapse (using average pooling), the sequence length is reduced to 1. An additional average pooling layer is applied to ensure the final output has the desired dimension.

$$X^u = \text{Collapse}(O^{(M)u}) \quad (13)$$

In this step,  $O^{(M)u}$  is the output of shape  $1 \times d_v$ .

### 4.3 Model Training and Optimization

For item ranking, we calculate the final score by taking the dot product of the **target items' embedding**  $Z_{target}^u$  obtained from the item encoder (Equation 7) and the transpose of the sequence embedding  $(X^u)^T$  as follows:

$$\hat{Y} = Z_{target}^u \cdot (X^u)^T \quad (14)$$

Given a positive item sampled from a user's sequence and a set of negative items drawn from a candidate pool (excluding items in the user's sequence), we train the model using the binary cross-entropy loss defined as follows:

$$\mathcal{L} = - \sum_{S \in \mathcal{S}} \left[ \log(\hat{Y}^{(+)}) + \sum_{n=0}^{N_{train}} (\log(1 - \hat{Y}_n^{(-)})) \right] \quad (15)$$

---

**Algorithm 1** Forward pass of the ConvRec model
 

---

**Require:** input encoded sequence  $Z^u$ , list of layer specs

$$M = [(kernel_1, stride_1), \dots, (kernel_m, stride_m)]$$

**Require:** activation function  $\sigma(\cdot)$ , pooling function  $\text{Pool}(\cdot)$

**Ensure:** logits  $\hat{Y}$

```

1:  $\text{ProgRes}^{(0)} \leftarrow Z^u$ 
2:  $O^{(0)u} \leftarrow Z^u$ 
3: for  $j = 1 \rightarrow M$  do
4:    $K^{(j)u} \leftarrow \text{1D-Conv}(\text{Padding}(O^{(j-1)uT}), kernel_j, \text{stride}_j)$  (Eq: 9)
5:    $G^{(j)u} \leftarrow \sigma(K^{(j)uT} W_G + b_G)$  (Eq: 10)
6:    $O^{(j)u} \leftarrow \text{LayerNorm}(G^{(j)u} + \alpha_1 \cdot \text{AvgPool}(Z^u) + \alpha_2 \cdot \text{ProgRes}^{(j-1)})$  (Eq: 11)
7:    $\text{ProgRes}^{(j)} \leftarrow O^{(j)u}$ 
8: end for
9:  $X^u \leftarrow \text{Collapse}(O^{(M)u})$  (Eq: 13)
10:  $\hat{Y} \leftarrow Z_{target}^u \cdot (X^u)^T$  (Eq: 14)
11: return  $\hat{Y}$ 
    
```

---

where  $\hat{Y}^{(+)}$  are the output scores for the positive sample and  $\hat{Y}_n^{(-)}$  are the output scores for  $N_{train}$  negative samples in our case we set  $N_{train}$  to 100.  $\mathcal{S}$  is the set of all sequences.

Many recommender models select the most recent part of the input sequence to represent the input and the shifted sequence to represent the positive items of the user [Kang and McAuley, 2018], [Rashed *et al.*, 2022]. While effective, this approach narrowly focuses on recent interactions, potentially overlooking valuable information present in earlier parts of the user’s history. In contrast, we adopt the training strategy proposed in [Seol *et al.*, 2024], where instead of always selecting the latest subsequence, a random subsequence of length  $L$  is sampled from the full user sequence  $S^u$ , and the final item in that sampled subsequence is used as the positive target. This allows the model to learn from various segments of the user’s behavior, not just the most recent interactions, and promotes better generalization.

## 5 Experiments

In this section we aim to answer the following research questions:

- RQ1: What is the model performance versus state-of-the-art methods?
- RQ2: How effective is each model component?
- RQ3: What is the effect of varying model hyperparameters such as kernel sizes and strides?
- RQ4: What are the computation and memory costs of the proposed model versus attention-based approaches?

**Datasets.** To comprehensively evaluate our proposed model, we conduct experiments on four widely used Amazon datasets. The Fashion and Men datasets [Elsayed *et al.*, 2022], [Rashed *et al.*, 2022], [Seol *et al.*, 2024] are derived from different segments of the Amazon fashion domain. The Fashion dataset includes six categories: men’s and

women’s tops, bottoms, and shoes. In contrast, the Amazon Men dataset comprises a broader range of subcategories, such as gloves, scarves, and sunglasses. For both datasets, item attributes are derived from visual features extracted using a pre-trained ResNet-50 model. In addition, we include the Amazon Games dataset [Rashed *et al.*, 2022], [Seol *et al.*, 2024], which contains video game items along with attributes such as category and brand. Lastly, the Amazon Beauty dataset [Rashed *et al.*, 2022], [Seol *et al.*, 2024] includes all items in the beauty category, where item attributes are also based on category and brand information. Table 1 summarizes the statistics of the datasets.

| Dataset | Users  | Items   | Interactions | Avg. Seq. | Max. Seq. | Attributes |
|---------|--------|---------|--------------|-----------|-----------|------------|
| Beauty  | 52,204 | 57,289  | 394,908      | 7         | 293       | 6507       |
| Games   | 31,013 | 23,715  | 287,107      | 9         | 860       | 506        |
| Fashion | 45,184 | 166,270 | 358,003      | 8         | 192       | 2048       |
| Men     | 34,244 | 110,636 | 254,870      | 7         | 308       | 2048       |

Table 1: Datasets statistics

**Evaluation Protocol.** We adopt a widely-used evaluation protocol based on the leave-one-out strategy [Kang and McAuley, 2018], [Rashed *et al.*, 2022], [Seol *et al.*, 2024]. Specifically, for each user, the last interacted item is held out for testing, the second-to-last item is used for validation, and the remaining sequence is used for training. For a fair comparison, we adopt the same negative sampling strategy used in prior studies [Rashed *et al.*, 2022], [Seol *et al.*, 2024], in which 100 negative items are sampled for evaluation.

We report standard top- $K$  metrics, Hit Rate (HR), and Normalized Discounted Cumulative Gain (NDCG) to effectively capture the performance of our model.

**Baselines.** We selected several state-of-the-art sequential and attribute-aware sequential recommendation models to assess the performance of our proposed method.

- BERT4Rec [Sun *et al.*, 2019]: A sequential recommendation framework based on a bidirectional Transformer, trained to predict randomly masked items within the input sequence.
- SASRec [Kang and McAuley, 2018]: A self-attention-based model that learns item dependencies in the sequence to generate next-item.
- CosRec [Yan *et al.*, 2019]: A 2D convolutional model that encodes item sequences to capture complex pairwise and high-order item interactions.
- TiSASRec [Li *et al.*, 2020]: An extension of SASRec that integrates temporal information (e.g., time intervals between interactions) into the attention mechanism.
- $S^3$ -Rec [Zhou *et al.*, 2020]: An attribute-aware sequential recommendation model that employs self-supervised pre-training to enhance item and user representations.
- SASRec++ [Rashed *et al.*, 2022]: An enhanced version of SASRec that incorporates item attribute information to improve recommendation accuracy.
- DIF-SR [Xie *et al.*, 2022]: A state-of-the-art approach that injects item features into the attention layers rather than restricting them to the input layer.

| Method                   | Beauty              |                       | Games               |                       | Fashion               |                       | Men                   |                       |
|--------------------------|---------------------|-----------------------|---------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
|                          | HR@10               | NDCG@10               | HR@10               | NDCG@10               | HR@10                 | NDCG@10               | HR@10                 | NDCG@10               |
| <b>BERT4Rec</b>          | 0.478               | 0.318                 | 0.705               | 0.509                 | 0.328                 | 0.209                 | 0.315                 | 0.193                 |
| <b>SASRec</b>            | 0.485               | 0.322                 | 0.742               | 0.541                 | 0.381                 | 0.245                 | 0.397                 | 0.259                 |
| <b>TiSASRec</b>          | 0.492               | 0.333                 | 0.748               | 0.533                 | 0.384                 | 0.234                 | 0.333                 | 0.194                 |
| <b>CosRec</b>            | 0.403               | 0.253                 | 0.662               | 0.444                 | 0.274                 | 0.150                 | 0.294                 | 0.164                 |
| <b>S<sup>3</sup>-Rec</b> | 0.538               | 0.371                 | 0.765               | 0.549                 | 0.367                 | 0.239                 | 0.365                 | 0.238                 |
| <b>SASRec++</b>          | 0.557               | 0.383                 | 0.752               | 0.533                 | 0.546                 | 0.344                 | 0.500                 | 0.315                 |
| <b>CARCA</b>             | 0.579               | 0.396                 | 0.782               | 0.573                 | 0.591                 | 0.381                 | 0.550                 | 0.349                 |
| <b>CARCA-LIP</b>         | 0.608               | 0.423                 | 0.762               | 0.560                 | 0.648                 | 0.427                 | <b>0.614</b>          | 0.398                 |
| <b>ProxyRCA</b>          | <b>0.609</b>        | <b>0.430</b>          | <b>0.805</b>        | <b>0.601</b>          | <b>0.661</b>          | <b>0.435</b>          | 0.613                 | <b>0.399</b>          |
| <b>ConvRec (ours)</b>    | <b>0.629 ± 1e-3</b> | <b>0.439 ± 3.5e-3</b> | <b>0.819 ± 1e-3</b> | <b>0.617 ± 2.5e-3</b> | <b>0.688 ± 2.6e-3</b> | <b>0.469 ± 3.8e-3</b> | <b>0.618 ± 1.2e-3</b> | <b>0.406 ± 1.3e-3</b> |
| Improv.(%)               | 3.28%               | 2.09%                 | 1.74%               | 2.66%                 | 4.08%                 | 7.82%                 | 0.65%                 | 2.00%                 |

Table 2: Model performance and comparison against baselines on four benchmark datasets using **100** negative test samples. The best results are reported in **red** and the second best in **blue**.

| Method                | Beauty       |              | Games        |              |
|-----------------------|--------------|--------------|--------------|--------------|
|                       | HR@10        | NDCG@10      | HR@10        | NDCG@10      |
| <b>BERT4Rec</b>       | 0.013        | 0.007        | 0.059        | 0.030        |
| <b>SASRec</b>         | 0.008        | 0.004        | 0.026        | 0.014        |
| <b>TiSASRec</b>       | 0.048        | <b>0.027</b> | 0.095        | 0.047        |
| <b>CosRec</b>         | 0.023        | 0.011        | 0.069        | 0.035        |
| <b>SASRec++</b>       | 0.027        | 0.013        | 0.075        | 0.037        |
| <b>DIF-SR</b>         | <b>0.051</b> | <b>0.025</b> | 0.109        | 0.048        |
| <b>CARCA-LIP</b>      | 0.037        | 0.017        | 0.116        | 0.064        |
| <b>ProxyRCA</b>       | 0.049        | <b>0.025</b> | <b>0.136</b> | <b>0.075</b> |
| <b>ConvRec (ours)</b> | <b>0.052</b> | <b>0.027</b> | <b>0.141</b> | <b>0.081</b> |
| Improv.(%)            | 1.9%         | 0.0%         | 3.6%         | 8.0%         |

Table 3: Model performance and comparison against baselines on the Beauty and Games datasets using **all items** as negative test samples. The best results are reported in **red** and the second best in **blue**.

- CARCA [Rashed *et al.*, 2022]: A context-aware recommendation method that employs cross-attention to compute item scores.
- CARCA-LIP [Seol *et al.*, 2024]: A CARCA model variant proposed in [Seol *et al.*, 2024] introduces an improved training approach and enhanced item embeddings, resulting in better performance.
- ProxyRCA [Seol *et al.*, 2024]: An advancement over CARCA that introduces proxy-based item embeddings, further improving item representation quality.

### 5.1 Model Performance Against Baselines

Table 2 reports the HR@10 and NDCG@10 of all methods on the four benchmark datasets, evaluated with 100 negative samples per user. For each dataset, the mean and standard deviation are reported based on three independent runs. The baseline models were trained and tuned according to the guidelines and hyperparameter settings in their original papers. ConvRec achieves the best results in both HR@10 and NDCG@10 metrics across all datasets. ProxyRCA is the strongest baseline and ranks second in terms of overall performance in most of the cases. Relative to ProxyRCA, ConvRec yields improvements ranging from 0.65% to 4.08% in HR@10 and from 2% to 7.82% in NDCG@10. In overall, attribute-aware models outperform sequence-only baselines. Among them, all recent top performers are attention-based models, whereas, to the best of our knowledge, ConvRec is the only convolution-based attribute-aware model.

Table 3 compares model performance on the Beauty

| Study         | Dataset      |              |              |              |
|---------------|--------------|--------------|--------------|--------------|
|               | Beauty       | Games        | Fashion      | Men          |
| w/o Intervals | 0.625        | 0.817        | 0.682        | 0.616        |
| w/o Residuals | 0.602        | 0.810        | 0.659        | 0.574        |
| w/ one Conv   | 0.617        | 0.796        | 0.674        | 0.605        |
| w/ AvgPool    | 0.602        | 0.804        | 0.669        | 0.611        |
| ConvRec       | <b>0.629</b> | <b>0.819</b> | <b>0.688</b> | <b>0.618</b> |

Table 4: Ablation studies on different model components (HR@10)

and Games datasets using the all-item evaluation setting. While baseline sequential models (e.g., BERT4Rec, SASRec) perform modestly, stronger baselines such as TiSASRec and ProxyRCA achieve competitive results. Our proposed ConvRec surpasses all existing methods, delivering the best HR@10 and NDCG@10 on both datasets. ConvRec achieves improvements of up to 3.6% in HR@10 and 8.0% in NDCG@10 over the best baseline, indicating its enhanced ability to model sequential dependencies and user-item interactions.

### 5.2 Ablation Studies

The ablation studies are organized into four main parts:

1. **Analyzing the impact of individual model components** on overall performance.
2. **Comparing memory and runtime efficiency** between the proposed convolution-based model and a multi-head attention-based model, such as ProxyRCA.
3. **Exploring alternative convolutional architectures** by testing different kernel sizes and strides. Additionally, testing the hyperparameter sensitivity of the model.
4. **Evaluating the effect of varying sequence lengths** on model performance, using the *Games* dataset, which has the longest average user sequence length.

#### Effect of Model Components

Table 4 presents the impact of various model components on overall performance. One key factor examined is the concatenation of the time intervals between successive interactions. This component shows a particularly notable impact on the Beauty dataset, highlighting the importance of temporal information in modeling user behavior.

Another important component is the residual connection. Its effect is evident on all datasets for example in the Men

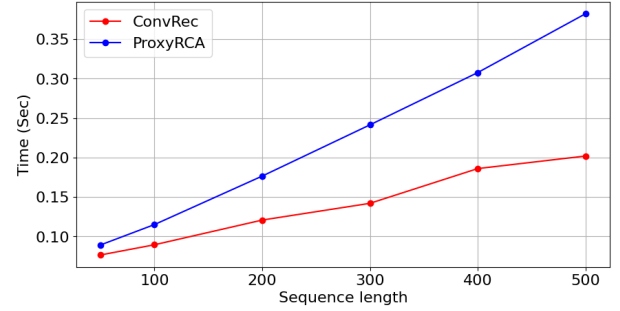
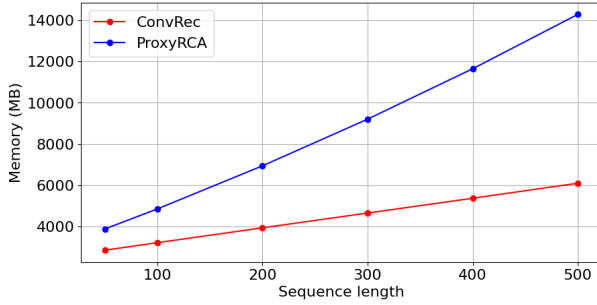


Figure 4: Comparison of memory usage and execution time between ConvRec and ProxyRCA models at batch size 128.

dataset, removing the two residual connections leads to a significant performance drop from 0.618 to 0.574. This underscores the value of incorporating residual connections into the model architecture.

Furthermore, we evaluate a variant of the model that uses a single convolutional layer with both kernel size and stride equal to the sequence length. This configuration effectively acts as a fully connected layer over the entire input sequence. While its performance is slightly lower than the proposed model, the drop is not substantial.

Additionally, we consider a baseline that computes the average pooling of item embeddings without any additional layers. This approach generally results in a performance decline across most datasets. However, the drop is relatively minor in the Men dataset, where the performance decreases to 0.611.

| (Kernel size, Stride)      | HR@10        | NDCG@10      |
|----------------------------|--------------|--------------|
| { (2, 2), (5, 5), (7, 7) } | <b>0.630</b> | <b>0.438</b> |
| { (3, 3), (5, 5), (5, 5) } | 0.625        | 0.435        |
| { (5, 5), (2, 2), (7, 7) } | 0.624        | 0.435        |
| { (7, 7), (5, 5), (2, 2) } | 0.622        | 0.432        |
| { (10, 10), (7, 7) }       | 0.621        | 0.430        |
| { (2, 1), (5, 5), (7, 7) } | 0.623        | 0.431        |
| { (2, 1), (5, 2), (7, 3) } | 0.615        | 0.421        |
| { (3, 1), (3, 1), (3, 1) } | 0.593        | 0.399        |

Table 5: Comparison between different kernel sizes, strides, and number of layers (Beauty dataset).

### Memory and Runtime Efficiency

We evaluated the training memory and runtime efficiency on Beauty dataset of our model and ProxyRCA using a batch size of 128 on NVIDIA A40 GPU. As shown in Figure 4, ProxyRCA exhibits a quadratic increase in memory consumption with respect to sequence length, rising from 3,878 MB for sequences of length 50 to 14,280 MB for sequences of length 500. In contrast, our model demonstrates linear memory growth, scaling from 2,849 MB to 6,096 MB under the same conditions.

Runtime performance follows a similar trend: ProxyRCA requires 0.0893 seconds per batch for sequences of length 50 and 0.382 seconds for length 500, while our model completes the same tasks in 0.076 and 0.202 seconds, respectively. These results highlight our model’s superior scalability in both memory efficiency and computational speed.

### Hyperparameter sensitivity

We explore alternative network architectures by experimenting with different kernel sizes. Furthermore, we analyze the impact of various hyperparameter settings, such as the embedding size and the dropout rate, to assess the model’s sensitivity to different configurations (Appendix A<sup>†</sup>). Lastly, we examined the performance across user groups with varying sequence lengths to evaluate the effectiveness of our convolution-based model compared to an attention-based model such as **ProxyRCA** (Appendix B<sup>†</sup>).

*Kernel size.* Since kernel size plays a crucial role in convolutional models, tuning it directly impacts the network architecture. As shown in Table 5, we experimented with various kernel sizes across different numbers of layers. The choice of kernel size for the first layer proved particularly important for the final model performance. The best results were obtained with a first-layer kernel size of 2, where each pair of items is aggregated together, followed by subsequent layers that aggregate the next 5 groups, and so on. Additionally, we experimented with different strides, such as a stride of 1 to combine each item with its preceding and succeeding items for a richer representation. However, this did not improve performance, and using a stride equal to the kernel size yielded the best results.

## 6 Conclusion

This paper introduces ConvRec, a convolution-based model for attribute and context-aware sequential recommendation. By leveraging adaptive convolution layers, the model compresses user historical sequences into compact yet highly expressive representations. Experimental results demonstrate that ConvRec outperforms state-of-the-art methods in recommendation accuracy. Moreover, compared to traditional multi-head attention approaches, the proposed model achieves superior efficiency in both memory usage and runtime, making it scalable for real-world applications.

### Contribution Statement

Shereen Elsayed and Ngoc Son Le contributed equally to this work.

## References

- [Cho *et al.*, 2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [Dang *et al.*, 2023] Yizhou Dang, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, Xiaoxiao Xu, Qinghui Sun, and Hong Liu. Ticoserec: Augmenting data to uniform sequences by time intervals for effective recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36(6):2686–2700, 2023.
- [Elsayed *et al.*, 2022] Shereen Elsayed, Lukas Brinkmeyer, and Lars Schmidt-Thieme. End-to-end image-based fashion recommendation. In *Workshop on Recommender Systems in Fashion and Retail*, pages 109–119. Springer, 2022.
- [Hendrycks and Gimpel, 2016] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [Hidasi and Karatzoglou, 2018] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 843–852, 2018.
- [Jannach and Ludewig, 2017] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 306–310, 2017.
- [Kang and McAuley, 2018] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.
- [Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [Li *et al.*, 2020] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pages 322–330, 2020.
- [Lin *et al.*, 2017] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.
- [Lin *et al.*, 2024] Xiaolin Lin, Jinwei Luo, Junwei Pan, Weiwei Pan, Zhong Ming, Xun Liu, Shudong Huang, and Jie Jiang. Multi-sequence attentive user representation learning for side-information integrated sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 414–423, 2024.
- [Liu *et al.*, 2021a] Chang Liu, Xiaoguang Li, Guohao Cai, Zhenhua Dong, Hong Zhu, and Lifeng Shang. Noninvasive self-attention for side information fusion in sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4249–4256, 2021.
- [Liu *et al.*, 2021b] Zhiwei Liu, Yongjun Chen, Jia Li, Philip S Yu, Julian McAuley, and Caiming Xiong. Contrastive self-supervised sequential recommendation with robust augmentation. *arXiv preprint arXiv:2108.06479*, 2021.
- [Liu *et al.*, 2022] Minhao Liu, Ailing Zeng, Muxi Chen, Zhijian Xu, Qiuxia Lai, Lingna Ma, and Qiang Xu. Scinet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 35:5816–5828, 2022.
- [Rashed *et al.*, 2022] Ahmed Rashed, Shereen Elsayed, and Lars Schmidt-Thieme. Context and attribute-aware sequential recommendation via cross-attention. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 71–80, 2022.
- [Rendle *et al.*, 2010] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.
- [Rendle *et al.*, 2012] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.
- [Seol *et al.*, 2024] Jinseok Seol, Minseok Gang, Sang-goo Lee, and Jaehui Park. Proxy-based item representation for attribute and context-aware recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 616–625, 2024.
- [Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of machine Learning research*, 6(Sep):1265–1295, 2005.
- [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 *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.
- [Tang and Wang, 2018] Jiayi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.
- [Wang *et al.*, 2020] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1101–1110, 2020.

- [Wang *et al.*, 2024] Shiyu Wang, Jiawei Li, Xiaoming Shi, Zhou Ye, Baichuan Mo, Wenze Lin, Shengtong Ju, Zhixuan Chu, and Ming Jin. Timemixer++: A general time series pattern machine for universal predictive analysis. *arXiv preprint arXiv:2410.16032*, 2024.
- [Xie *et al.*, 2022] Yueqi Xie, Peilin Zhou, and Sunghun Kim. Decoupled side information fusion for sequential recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 1611–1621, 2022.
- [Xu *et al.*, 2019] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng S. Sheng, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. Recurrent convolutional neural network for sequential recommendation. In *The world wide web conference*, pages 3398–3404, 2019.
- [Xu *et al.*, 2025] Zitao Xu, Xiaoqing Chen, Weike Pan, and Zhong Ming. Heterogeneous graph transfer learning for category-aware cross-domain sequential recommendation. In *THE WEB CONFERENCE 2025*, 2025.
- [Yan *et al.*, 2019] An Yan, Shuo Cheng, Wang-Cheng Kang, Mengting Wan, and Julian McAuley. Cosrec: 2d convolutional neural networks for sequential recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2173–2176, 2019.
- [Zhou *et al.*, 2020] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1893–1902, 2020.