AdaMCT: Adaptive Mixture of CNN-Transformer for Sequential Recommendation

Juyong Jiang∗†
The Hong Kong University of Science and Technology (Guangzhou)
jjiang472@connect.hkust-gz.edu.cn

Peiyuan Zhang∗
The Hong Kong University of Science and Technology
pzhangao@cse.ust.hk

Yingtao Luo
Carnegie Mellon University
yingtaoluo@cmu.edu

Chaozhuo Li‡
Microsoft Research Asia
cli@microsoft.com

Kai Zhang‡
East China Normal University
kzhang@cs.ecnu.edu.cn

Senzhang Wang
Central South University
szwang@csu.edu.cn

Xing Xie
Microsoft Research Asia
xing.xie@microsoft.com

Sunghun Kim
The Hong Kong University of Science and Technology (Guangzhou)
hunkim@ust.hk

ABSTRACT
Sequential recommendation (SR) aims to model users’ dynamic preferences from a series of interactions. A pivotal challenge in user modeling for SR lies in the inherent variability of user preferences. An effective SR model is expected to capture both the long-term and short-term preferences exhibited by users, wherein the former can offer a comprehensive understanding of stable interests that impact the latter. To more effectively capture such information, we incorporate locality inductive bias into the Transformer by amalgamating its global attention mechanism with a local convolutional filter, and adaptively ascertain the mixing importance on a personalized basis through layer-aware adaptive mixture units, termed as AdaMCT. Moreover, as users may repeatedly browse potential purchases, it is expected to consider multiple relevant items concurrently in long-/short-term preferences modeling. Given that softmax-based attention may promote unimodal activation, we propose the Squeeze-Excitation Attention (with sigmoid activation) into SR models to capture multiple pertinent items (keys) simultaneously. Extensive experiments on three widely employed benchmarks substantiate the effectiveness and efficiency of our proposed approach. Source code is available at https://github.com/juyongjiang/AdaMCT.

CCS CONCEPTS
• Information systems → Recommender systems.

KEYWORDS
Sequential Recommendation; CNNs; Transformer

ACM Reference Format:
Juyong Jiang, Peiyuan Zhang, Yingtao Luo, Chaozhuo Li, Jae Boum Kim, Kai Zhang, Senzhang Wang, Xing Xie, and Sunghun Kim. 2023. AdaMCT: Adaptive Mixture of CNN-Transformer for Sequential Recommendation. In Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM ’23), October 21–25, 2023, Birmingham, United Kingdom. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583780.3614773

1 INTRODUCTION
Recommender systems facilitate the provision of tailored information to users, taking into account their preferences as evidenced by historical interactions [12], which are widely applied in e-commerce websites, web searches, and so forth [10, 67]. Analogous with words of sentences, a user’s interactions with items naturally constitute a behavioral sequence. Thus, sequential recommendation (SR) has been proposed to encapsulate the sequential characteristics of user behaviors, thereby modeling users’ preferences more effectively [19, 25].

A vital challenge of user modeling for SR lies in the inherent variability of user preferences. In large-scale recommender systems, millions of new items are generated and incorporated into the candidate pool on a daily basis [65]. It is imperative for recommender systems to promptly and accurately discern users’ short-term preferences, as their attention is readily captivated by emerging trends and popular topics. In contrast, users’ long-term preferences offer a valuable insight into their aggregated, stable interests, effectively complementing their short-term interests. Hence, it is crucial for SR models to adeptly capture both the dynamic nature of short-term preferences and the enduring stability of long-term preferences exhibited by users.

Fortunately, recent studies in SR have endeavored on this target by modeling and integrating long- and short-term preferences prior to making recommendations. The initial line of research [9, 50, 64]
concentrates on capturing long-term preferences while employing the most recently interacted item as a representation of short-term preferences. However, due to the SR-related inductive bias, specifically local invariance [4], which suggests that the local order of the last few items is inconsequential, it becomes arduous to comprehensively represent a user’s short-term interests using only the final item. Furthermore, these works merely fuse the long-/short-term preferences through summation or concatenation [4, 18, 21, 33, 57, 58, 60]. As a result, these approaches fail to obtain personalized weights for long-/short-term preferences pertaining to different items for each user.

Another line of works adopts a parallel paradigm to model the long-/short-term preferences respectively with recurrent and attentive models [23, 37, 69] or even explicit user behavior graphs [55]. Subsequently, they implement a gating mechanism [11] to facilitate the integration of personalized long-term and short-term preferences. Nevertheless, the long-/short-term preference modeling processes are segregated. Given that users’ long-term preferences can offer a comprehensive understanding of stable interests that influence short-term interests and vice versa [1, 8, 61], this paradigm may adversely affect the comprehension of users’ overall preferences. Therefore, it is reasonable that the long-/short-term preferences should be mutually reinforced in the context of user modeling.

In particular, modeling the long-/short-term preferences in SR is non-trivial, primarily encountering the following three challenges: (1) How can long-term and short-term preferences be modeled conjointly? As users’ long-term preferences could provide users’ aggregated stable interests that influence the short-term interests and vice versa [55], these preferences should engage in a mutual reinforcement within user modeling. (2) How to integrate the short-term and long-term preferences effectively? Both types of interests play a crucial role in real-world recommendations, despite the inherent biases between them. Models must be capable of discerning which interest is more pertinent in varying situations. Previous studies [4, 21, 55, 57] have relied on simple summation, concatenation, or a singular gating mechanism, leading to restricted model capacity. (3) How can multiple relevant items be considered concurrently in long-term and short-term preference modeling? As illustrated in Figure 1, users may repeatedly browse and compare potential purchases, resulting in more than one item being highly correlated with their preferences. However, the softmax activation in the attention scheme necessitates that the sum of all item scores equals 1, creating mutually exclusive relations [22] that impede the simultaneous activation of multiple highly-related items. These three challenges are of significant importance in real-world scenarios, yet there is a scarcity of research that systematically addresses them in unison.

To address these challenges, we introduce a novel Adaptive Mixture of CNN-Transformer for Sequential Recommendation (AdaMCT) that explicitly learns personalized behavior preferences to adaptively leverage locality inductive bias (CNNs) [48] and global interactions (Transformer) [28, 46] modeling. Specifically, we employ a local-global dependencies mechanism comprising both (local) convolutional layers and (global) self-attention layers, facilitating the joint modeling of long-term and short-term preferences. The CNN layer exhibits heightened sensitivity to short-term interest patterns, while the self-attention layer is better suited for identifying long-term dependency patterns. To effectively integrate long-term and short-term preferences, we devise adaptive mixture units that decouple the fusion process into multiple layers, thereby enhancing expressibility and adaptively aggregating long-term and short-term preferences for personalized user modeling. Furthermore, we propose the Squeeze-Excitation Attention to generate attention scores for both short-term and long-term information branches. Squeeze-Excitation Attention eliminates the sum-up-to-1 constraint, allowing for the consideration of multiple relevant purchases simultaneously and enhancing model expressibility. We validate the AdaMCT approach on three public benchmarks. Substantial improvements across multiple indicators demonstrate the efficacy of the proposed methods. The main contributions of this paper are as follows:

- We incorporate the locality inductive bias (CNNs) into the Transformer structure to model the long-/short-term preferences conjointly, and devise the adaptive mixture units that decouple the fusion process into multiple layers, thereby enhancing expressibility and adaptively aggregating long-term and short-term preferences for personalized user modeling.
- We propose Squeeze-Excitation Attention (SEAtt) to replace the softmax operation in both local and global information branches to simultaneously take into account multiple relevant purchases to effectively improve the representation learning of sequential dependencies.
- Extensive experiments on three public datasets demonstrate that our proposed solution achieves comparable (or better) performance than the previous Transformer and CNNs-based models. Ablation studies show the efficacy of our proposed components.

2 RELATED WORK

2.1 Sequential Recommendation

Sequential recommendation aims to predict the future item or item list based on a historical user behavior sequence. In the literature, early works mainly use Markov Chain models [16, 29, 43] that make personalized next recommendations conditioned on the previous one’s behavior. These methods only capture adjacent local
sequential patterns. Later on, the genre of neural network-based models starts to rise. Recurrent Neural Networks (RNNs) improve the recommendation performance in many tasks with sequential dependencies and memory mechanisms [7, 35, 36, 59]. RNNs impose strict sequential order in representation learning and face the global dependency problem when the sequence is too long. Meanwhile, Convolutional Neural Networks (CNNs) are used to capture local features across the sequence, emphasizing the impact of more recent behaviors [48, 63, 68]. More recently, attentive modules are extensively studied in the sequential recommendation venue, resulting in extraordinary performances [27, 28]. Self-attention models allow point-to-point feature interaction within the item sequence, which solves the global dependency problem and uses a longer sequence with more information [26, 28, 39]. Meanwhile, graph neural networks have been attempted to discover the graph-based topological interaction of sequences [30, 41, 51, 54]. Transformer and Bidirectional Transformer (BERT) that combine multiple blocks including attention layers, feed-forward networks, and position embeddings also report state-of-the-art performances in sequential recommendation tasks [3, 46, 53]. Mixed models [31, 57, 62] that consider both global and local preferences have also received extensive attention in recent years.

2.2 Local-Global Awareness

Local-global awareness is a topic that has raised discussions in the venue of recommendation systems. Many prior works [24, 40, 45] propose and design local-global memory mechanisms to learn different-level item sequences, while they use a simple concatenation of neural modules without an adaptive learnable preference factor. However, some works [18, 32] also consider the adaptive mixture of local-global representation learning and show some advantages, but the difference from previous works is that we first attempt to adopt CNNs and Transformer to capture local and global dependencies in SR. Moreover, Local-global regularized is used to learn the local and global item correlations with both deep neural network and matrix factorization [34, 66]. These works design user-specific strategies to fuse local and global information into the recommendation model [6, 47]. Other works focus on improving neural architectures. Global and local sparse linear method models are combined to make recommendations for different user preferences [21]. A recurrent convolutional network is proposed to learn compact global sequential dependency with RNNs and extract local dependency using CNNs [57], which marks a combined use of sequence order bias and locality bias. An encoder for modeling the global cross-session dependency is proposed to improve the self-attention model [58]. Local generative models that capture different user preferences are combined to make a global prediction [33]. Local invariance is considered in session-based recommendation [4] as extra local information to substitute the main model. Attention and graph network are combined to model different user interests [60] with graph topology inductive bias. Recently, some papers [38, 52, 62] find that the personalized and adaptive use of model components can effectively improve the model performance on RNNs. However, most works combining multiple neural modules, especially the state-of-the-art Transformer, still rarely consider the user-specific learnable adaptation. Moreover, they hardly consider the comprehensive inductive biases from locality to global dependency, and from sequential order to positional equivalence.

3 METHODOLOGY

3.1 Problem Statement

\[ \mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\} \] denotes users; \( \mathcal{V} = \{v_1, v_2, ..., v_{|\mathcal{V}|}\} \) denotes items, and \( S_{1:n}^{(u)} = [v_1^{(u)}, v_2^{(u)}, ..., v_n^{(u)}] \) denotes the interaction sequence in chronological order of user \( u \in \mathcal{U} \). Each \( v_i^{(u)} \in \mathcal{V} \) refers to an item that \( u \) has interacted with at time step \( i \). Sequential recommendation aims to predict the item that \( u \) will interact with at the \( (n+1) \text{th} \) timestamp. The conditional probability of each item \( v \in \mathcal{V} \) becoming the \( (n+1) \text{th} \) item is denoted as \( \mathcal{P}_{\theta}(o_{n+1}^{(u)} = v | S_{1:n}^{(u)}) \), where \( \theta \) represents the model parameters.

3.2 Model Architecture

The overall model architecture is shown in Figure 2. Our proposed AdaMCT consists of 1) an embedding module that learns the dense representations of items, including item embeddings and positional embeddings; 2) a multi-stacked Adaptive Mixture of CNN-Transformer blocks, in which both comprise a global attention module (Transformer) that learns the global user interest preferences with squeeze-excitation attention (SEAtt), a local convolutional module (CNNs) that learns user local interest pattern with SEAtt, and a module- and layer-aware adaptive mixture units, that makes a personalized balanced use of local and global dependencies modules; and 3) an output layer with cross-entropy loss that computes the matching probability.

3.3 Embedding Module

The embedding module contains an item embedding layer and a positional embedding layer. The item embedding layer \( E_{item} \in \mathbb{R}^{|\mathcal{V}| \times d_{model}} \) with the latent dimension of \( d_{model} \) encodes each scalar item ID \( v_i^{(u)} \in \mathbb{R} \) into a dense latent vector representation \( e_i^{(u)} \in \mathbb{R}^{d_{model}} \), aiming to reduce computation and improve representation learning. The positional embeddings \( P_{pos} \in \mathbb{R}^{n \times d_{model}} \) follow [28] to employ learnable embeddings and maintain the same embedding dimensions of the item embedding. The overall embedding module transforms the sequence of \( S_{1:n}^{(u)} = [v_1^{(u)}, v_2^{(u)}, ..., v_n^{(u)}] \in \mathbb{R}^n \) into \( E_{item}(S_{1:n}) = [E_{item}(v_1), E_{item}(v_2), ..., E_{item}(v_n)] \in \mathbb{R}^{n \times d_{model}} \).

3.4 Adaptive Mixture of CNN-Transformer

AdaMCT has three sub-layers, a global attention module (Transformer) with SEAtt referred to as \( \mathcal{G}_{Trm} \), a local convolutional module (CNNs) with SEAtt referred to as \( \mathcal{L}_{Conv} \), and adaptive mixture units referred to as \( \mathcal{M}_{Mix} \), as shown in the middle of Figure 2.

3.4.1 Global Attention Module.

As shown in Figure 2, we first utilize a linear layer, followed by layer normalization [2], to adjust a linear dimensionality and enhance the non-linear representation. Then, a multi-head self-attention layer (MHSAs) followed by layer normalization is adopted to capture global dependencies of sequence. Finally, a squeeze-excitation attention layer (SEAtt) to re-weight the importance of each item from the sequence level.
Formally, \( \mathcal{G}^{\text{Trm}} = \text{SEAtt} (\text{MHSA}(E_{\text{global}}(H_{u}^{i-1})) \otimes \text{MHSA}(E_{\text{global}}(H_{u}^{i}))) \),

This equation formalizes the SEAtt module, which takes as input the global and local self-attention modules and outputs a new representation. The self-attention module is applied to the concatenation of the two input representations, effectively enabling the model to learn long-range dependencies within the input sequence.

The equation captures the essence of the SEAtt module, where the SEAtt function combines two self-attention modules, \( \text{MHSA}(E_{\text{global}}(H_{u}^{i-1})) \) and \( \text{MHSA}(E_{\text{global}}(H_{u}^{i})) \), through a concatenation operation. This allows the model to incorporate both global and local contextual information in a single operation, enhancing its ability to capture nuanced relationships within the input sequence.

In essence, the SEAtt module serves as a bridge between the global and local attention mechanisms, enabling the model to leverage both perspectives effectively.

By utilizing the SEAtt module, the model can achieve a more balanced understanding of the input sequence, striking a harmonious balance between capturing global patterns and local subtleties, which is crucial for tasks requiring comprehensive semantic comprehension.

This dual-layered approach not only enhances the model’s interpretability but also significantly boosts its performance on tasks involving complex data inputs, making it a powerful tool in the realm of natural language processing and machine learning.
\[ C^i_j = \Phi(E_{\text{en}}^{\text{local}}(H^{i-1}_{(u)}|j+k-1|F^i)), \]

where \( \Phi(\cdot) \) denotes the activation function of convolutional layers in which we use the ReLU by default. \( E_{\text{en}}^{\text{local}}(H^{i-1}_{(u)}|j+k-1|) \in \mathbb{R}^{k \times d_{\text{model}}} \) represents the sub-sequence with indexes from \( j \) to \( j+k-1 \). \( \text{Conv}^{k \times 3}(E_{\text{en}}^{\text{mid}}(H^{i-1}_{(u)})) \in \mathbb{R}^m \) denotes the convolutional value in the position \( j \).

Furthermore, we input the above output of convolutional layer to the SEAtt module to re-weight the importance of each item from the sequence level. Formally,

\[ \mathcal{L}_{\text{Conv}} = \text{SEAtt}(\text{Conv}^{k \times 3}(E_{\text{en}}^{\text{mid}}(H^{i-1}_{(u)}))) \otimes \text{Conv}^{k \times 3}(E_{\text{en}}^{\text{mid}}(H^{i-1}_{(u)})), \]

where \( \mathcal{L}_{\text{Conv}} \in \mathbb{R}^{n \times m} \) represents the output of local convolutional module. In this work, we set \( m = d_{\text{model}} \) to make \( \mathcal{L}_{\text{Conv}} \) have the same dimension as \( \mathcal{G}_{\text{Trm}} \), i.e., \( \mathcal{L}_{\text{Conv}} \in \mathbb{R}^{n \times d_{\text{model}}} \).

### 3.4.4 Adaptive Mixture Units

This module is shown in the top part of the AdaMCT in the middle of Figure 2. The adaptive mixture units adjust the personalized trade-off between the global attention module \( \mathcal{G}_{\text{Trm}} \) and the local convolutional module \( \mathcal{L}_{\text{Conv}} \). Specifically, it first leverages the average pooling operator to produce dimension-wise statistics, and then a linear layer with \( \Omega \) activation function is employed to get the adaptive coefficient for each user.

\[ \text{Ada}(u)(H^{i-1}_{(u)}) = \Omega(\text{Linear}_{\text{mid}}(\text{Pool}(H^{i-1}_{(u)}))), \]

\[ \text{Pool}(\text{Out}(H^{i-1}_{(u)})) = \frac{1}{n} \sum_{i=1}^{n} \text{Out}(H^{i-1}_{(u)}), \]

where \( \text{Pool}(\text{Out}(H^{i-1}_{(u)})) \in \mathbb{R}^{d_{\text{model}}} \) denotes user preference representation at \((i-1)\)-th layer. \( \Omega(\cdot) \) denotes the Sigmoid activation function which is used to scale the value to the range 0 to 1. \( H^{i-1}_{(u)} \) represents the output of \((i-1)\)-th layer. \( \text{Ada}(u)(H^{i-1}_{(u)}) \in \mathbb{R} \) is a personalized coefficient changing over different users since user preference representation is formulated by the average pooling of historical user behaviors. Thus, if we have \( |\mathcal{U}| \) users, it will produce \(|\mathcal{U}| \) coefficient values. Besides, the adaptive coefficient values are unique per the AdaMCT layer, which further extends the capacity of representation learning. In all, adaptive mixture units are module- and layer-aware with personalized inductive bias.

Finally, we mix the output of global attention module (\( \mathcal{G}_{\text{Trm}} \)) and local convolutional module (\( \mathcal{L}_{\text{Conv}} \)) with the adaptive coefficient (\( \text{Ada}(u) \)) to produce local-global dependencies representation.

\[ \mathcal{M}_{\text{Trm}} = \text{Ada}(u)(H^{i-1}_{(u)}) \odot \mathcal{L}_{\text{Conv}} + (1 - \text{Ada}(u)(H^{i-1}_{(u)})) \odot \mathcal{G}_{\text{Trm}} \]

where \( \text{Ada}(u)(H^{i-1}_{(u)}) \) and \( 1 - \text{Ada}(u)(H^{i-1}_{(u)}) \) are denoted by \( \alpha \) and \( \beta \) respectively in Figure 2. Furthermore, to maintain the dimension invariant, we apply a linear layer to adjust the latent dimension of \( \mathcal{M}_{\text{Trm}} \) and leverage a residual connection [14] around the input to make the \((i-1)\)-th layer’s output \( H^i_{(u)} \in \mathbb{R}^{n \times d} \) followed by layer normalization.

\[ H^i_{(u)} = \text{LN}(H^i_{(u)}) + \text{Dpt}(\text{Linear}_{\text{out}}(\mathcal{M}_{\text{Conv}}^i)) \]

#### 3.5 Output Layer

After \( L \) layers hierarchically aggregate information across all positions, we get the final output \( H^L_{(u)} \in \mathbb{R}^{n \times d_{\text{model}}} \) for all items of the input sequence. Then, we use the contextual hidden representation at the \( n \)-th time step of \( H^L_{(u)} \) as the input of the prediction layer, which consists of a two-layer feed-forward network and a softmax function to produce the full-ranked items probability distribution over the next items.

\[ P_{\text{out}}(e^u_{(n+1)}) = \sigma(\delta(H^L_{(u)}|n|W^{\text{Pred}} + b^{\text{Pred}})E^T_{\text{table}} + b^O), \]

where \( W^{\text{Pred}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}} \) and \( b^{\text{Pred}} \in \mathbb{R}^{d_{\text{model}}} \) are parameters and \( E^T_{\text{table}} \in \mathbb{R}^{d_{\text{model}} \times |V|} \) is an item embedding table. We follow [28, 46] to use a shared item embedding table of the embedding module to avoid overfitting and speed up model convergence.
Table 1: Dataset statistics after preprocessing. ‘#Inter.’ denotes the number of interaction, ‘#Len’ denotes average actions of Users/Items.

| Datasets | #Inter. | #Users | #Items | #Len | #LenI | Sparsity |
|----------|---------|--------|--------|------|-------|----------|
| Toys     | 167,597 | 19,412 | 12,101 | 8.6  | 16.4  | 99.93%   |
| Sports   | 296,337 | 35,598 | 18,357 | 8.3  | 16.1  | 99.95%   |
| Beauty   | 198,502 | 22,363 | 12,101 | 8.9  | 16.4  | 99.93%   |

3.5.1 Model Training. In this work, we adopt cross-entropy loss to supervised model training. Formally, the objective loss is formulated as follows:

$$L_{ce} (\theta) = - \sum_{u \in U} \log p_{\theta}(y_{n+1} = v | s_{t:n}^{(u)}) = - \sum_{u \in U} \log P_{out}(\hat{v}_{n+1}).$$

where $\theta$ represents the model parameters.

4 EXPERIMENTS

In this section, we present the empirical evaluations AdaMCT on three widely used benchmarks with varied domains and statistics, aiming to answer the following research questions.

- **RQ1**: Can our proposed AdaMCT outperform the state-of-the-art baselines for sequential recommendation?
- **RQ2**: What is the influence of various component designs in AdaMCT?
- **RQ3**: How do different hyper-parameters affect AdaMCT?
- **RQ4**: How efficient is AdaMCT compared to the baselines? How about the model complexity of AdaMCT?
- **RQ5**: Does the visualization of attention matrices of Transformer provide some interpretability for feature representations in the AdaMCT framework?

4.1 Experimental Settings

4.1.1 Dataset. In this work, our experiments are conducted on three real-world and widely used benchmarks.

- **Amazon Toys, Sports, Beauty**: This is a series of datasets [42] that consist of large corpora of product reviews crawled from Amazon.com. The datasets are split by the top-level product categories. We adopt the “Toys and Games”, “Sports and Outdoors”, and “Beauty” categories.

The selected datasets are the ones used in important prior works [5, 15, 28, 31, 46], from which we followed the same data preprocessing procedures. In essence, we convert each dataset into an implicit dataset by treating each rating or review as a presence of a user-item interaction. After that, we group the interactions by user IDs and sort them by timestamps to form a sequence for each user. We follow the same practice as in [5, 15, 28, 44, 46, 49] to discard short sequences less than five interactions. Table 1 describes the statistics of the datasets. These three datasets are frequently used as public benchmarks and differ in size and density, which gives us the opportunity to test models on different kinds of datasets.

4.1.2 Evaluation Metrics. To evaluate the performance of each method, we employ the widely used leave-one-out evaluation task for each user’s item sequence, which means we hold out the last item for the test, the second last item for validation, and the rest for training. Besides, we follow the common practice in [5, 28, 31, 46] to pair each ground truth item in the test set with 100 randomly sampled negative items that have not been interacted by the user. To make the task more realistic, we sample the negative items by uniform sampling. To score the ranked list, we employ two common evaluation metrics: Hit Ratio (HR), and Normalized Discounted Cumulative Gain (NDCG). Since we only have one ground truth item for each user, HR@K is equivalent to Recall@K. Therefore, both Recall@K and NDCG@K have large values when the ground truth item is ranked higher up in the top-K list. We report Recall and NDCG with $k = 1, 5, 10$.

4.1.3 Baseline Models. We choose some typical state-of-the-art SR methods for comparison, especially including individual CNNs (CaseNet, NextitNet) and Transformer (SASRec, BERT4Rec, GCSAN, and CORE). The baselines are introduced as follows.

- **GRU4Rec** [19]: A session-based recommender system based on recurrent neural network.
- **CaseNet** [48]: A CNN-based model with horizontal and vertical convolutional layers for personalized recommendation.
- **NextitNet** [63]: A simple convolutional generative network for next item recommendation.
- **SASRec** [28]: A Transformer-based model that uses self-attention network for the sequential recommendation.
- **BERT4Rec** [46]: A bidirectional Transformer-based model for the sequential recommendation.
- **HGN** [40]: A hierarchical gating networks for sequential recommendation.
- **GCSAN** [56]: A graph contextualized self-attention network for session-based recommendation.
- **SINE** [47]: A sparse-interest network for sequential recommendation.
- **CORE** [20]: A simple and effective session-based recommendation with unifying representation space.

4.1.4 Implementation Details. We evaluate baseline models based on the famous open-source RecBole framework2. We apply grid search to find the optimal hyper-parameters for each model. We consider hidden dimension size $d_{model} \in \{16, 32, 64, 96, 128\}$, number of heads $h \in \{2, 4, 8, 16, 32\}$, reduction ratio $r \in \{1, 2, 5, 10, 25\}$, kernel size $k \in \{3, 5, 7, 9, 11\}$, activation function $\Phi \in \{GELU, RelU', Swish', TanH', Sigmoid'\}$, hidden dropout ratio $d_1 \in [0, 1]$, attention dropout ratio $d_2 \in [0, 1]$ and learning rate $lr \in [0, 1]$. We report the results of each baseline under its optimal hyper-parameter settings. For our AdaMCT, the optimal hyper-parameters for each benchmark are Toys: $\{L = 2, k = 3, r = 2, \Phi = 'GELU', h = 4, d_{model} = 32, d_1 = 0.50, d_2 = 0.50, lr = 0.001\}$, Beauty: $\{L = 2, k = 5, r = 2, \Phi = 'Swish', h = 2, d_{model} = 16, d_1 = 0.43, d_2 = 0.57, lr = 0.02\}$, and Sports: $\{L = 2, k = 3, r = 5, \Phi = 'GELU', h = 4, d_{model} = 64, d_1 = 0.76, d_2 = 0.51, lr = 0.001\}$, respectively. We train the model using the Adam algorithm with $\beta_1 = 0.9, \beta_2 = 0.999$, and $\beta_2$ weight decay of 0.0, and linear decay of

1http://snap.stanford.edu/data/amazon/

2https://github.com/RUCAIBox/RecBole
Table 2: Comparison of recommendation performance with baseline models in numerical values. Bold scores are the best in each row, while underlined scores are the second best. * denotes CNN-based and Transformer-based models, respectively.

| Dataset | Metric        | (a) GRU4Rec | (b) Caser* | (c) NextItNet* | (d) SASRec | (e) BERT4Rec* | (f) HGN | (g) GCSAN | (h) SINER | (i) CORE | (j) AdaMTC | (k) Improv. |
|---------|---------------|-------------|------------|---------------|------------|---------------|--------|----------|----------|---------|-----------|------------|
| Toys    | Recall@1      | 0.1650      | 0.1264     | 0.1092        | 0.2120     | 0.1298        | 0.1679 | 0.1840   | 0.1745   | 0.1785  | 0.2184    | +3.02%     |
|         | Recall@5      | 0.3535      | 0.3144     | 0.2893        | 0.3956     | 0.3040        | 0.3609 | 0.3530   | 0.3678   | 0.3812  | 0.4132    | +4.45%     |
|         | Recall@10     | 0.4591      | 0.4231     | 0.3952        | 0.4886     | 0.4097        | 0.4683 | 0.4469   | 0.4665   | 0.4860  | 0.5088    | +4.13%     |
|         | NDCG@5        | 0.2630      | 0.2237     | 0.2019        | 0.3080     | 0.2195        | 0.2679 | 0.2718   | 0.2748   | 0.2842  | 0.3203    | +3.99%     |
|         | NDCG@10       | 0.2971      | 0.2588     | 0.2361        | 0.3380     | 0.2535        | 0.3024 | 0.3021   | 0.3066   | 0.3180  | 0.3511    | +3.88%     |
| Beauty  | Recall@1      | 0.1848      | 0.1503     | 0.1580        | 0.2154     | 0.1457        | 0.1835 | 0.1832   | 0.1903   | 0.1826  | 0.2178    | +1.11%     |
|         | Recall@5      | 0.3688      | 0.3364     | 0.3560        | 0.4014     | 0.3176        | 0.3793 | 0.3629   | 0.3864   | 0.3949  | 0.4207    | +4.81%     |
|         | Recall@10     | 0.4687      | 0.4370     | 0.4605        | 0.4933     | 0.4163        | 0.4792 | 0.4554   | 0.4806   | 0.4951  | 0.5178    | +5.48%     |
|         | NDCD@5        | 0.2806      | 0.2469     | 0.2611        | 0.3127     | 0.2348        | 0.2859 | 0.2771   | 0.2937   | 0.2938  | 0.3241    | +3.65%     |
|         | NDCD@10       | 0.3129      | 0.2793     | 0.2949        | 0.3424     | 0.2666        | 0.3181 | 0.3069   | 0.3239   | 0.3262  | 0.3554    | +3.80%     |
| Sports  | Recall@1      | 0.1423      | 0.1239     | 0.1255        | 0.1716     | 0.1178        | 0.1593 | 0.1369   | 0.1641   | 0.1453  | 0.1862    | +8.51%     |
|         | Recall@5      | 0.3386      | 0.3173     | 0.3275        | 0.3760     | 0.3036        | 0.3657 | 0.3303   | 0.3751   | 0.3713  | 0.4097    | +8.96%     |
|         | Recall@10     | 0.4606      | 0.4417     | 0.4578        | 0.4990     | 0.4269        | 0.4890 | 0.4457   | 0.4953   | 0.4993  | 0.5412    | +8.46%     |
|         | NDCG@5        | 0.2434      | 0.2226     | 0.2287        | 0.2775     | 0.2128        | 0.2660 | 0.2365   | 0.2728   | 0.2616  | 0.3019    | +8.79%     |
|         | NDCG@10       | 0.2828      | 0.2627     | 0.2708        | 0.3172     | 0.2525        | 0.3059 | 0.2737   | 0.3110   | 0.3010  | 0.3444    | +8.58%     |

The learning rate. To ensure fairness, we used the same maximum sequence length (N = 50 for Amazon) and the number of layers (L = 2) as in [28]. We train each model with early stop strategies until the validation accuracy does not improve for 20 epochs on a single NVIDIA A100 SXM4 80GB GPU with a batch size of 2048. The results of the test sets are reported in the following section.

4.2 Overall Performance (RQ1)

Table 2 shows the recommendation performance of our model and baselines on the three datasets. Here, we used the averaged performance run by five random seeds. Consequently, the improvement provided by AdaMCT is statistically and consistently significant over all datasets.

We highlight two important baselines Caser and SASRec which belong to the individual architecture of CNNs and Transformer respectively. AdaMCT surpasses Caser by a margin exceeding 4% on the Toys and Beauty datasets, and even over 8% on the Amazon Sports dataset, considering all metrics on average. Caser performs fairly well compared to GRU4Rec, implying that models containing local inductive bias are nontrivial in predicting the next item.

Self-attention based models such as SASRec and BERT4Rec show better performances over the RNN-based GRU4Rec and CNN-based Caser. This suggests that global attentive representation learning is relatively more important than the use of only one inductive bias. Meanwhile, GNN-based models, i.e., GCSAN outperforms RNN-based and CNN-based models, underscoring the importance of capturing pairwise relationships between items. Nevertheless, these models fall short of surpassing SASRec, indicating that sequential relationships prevail when modeling user preferences in sequential recommendation.

On average, AdaMCT outperforms SASRec across all three datasets. As such, AdaMCT emerges as the most effective model, adeptly considering both local and global dependencies of user interest patterns in an adaptive manner. If we extend the comparison to ablated AdaMCT, we find that AdaMCT performs better than SASRec while allowing either the local module or the global module to be adaptive, as shown in Table 3. This means that with our proposed adaptive mixture units, only CNN-based components can already be comparable with SASRec. On the contrary, when we...
Figure 3: Performance (NDCG@10) comparison of with(w) or without(w/o) proposed components including a global attention module (Transformer) $G_{Trm}$, a local convolutional module (CNNs) $L_{Conv}$, an adaptive mixture unit $A_{Mix}$, and two squeeze-excitation attention (SEAtt) per layer of AdaMCT on three datasets.

Table 5: Performances (NDCG@10) comparison of Squeeze-Excitation Attention (SEAtt) with local (CNNs) $L_{Conv}$ and global (Transformer) $G_{Trm}$ awareness on three datasets. ‘♦’ and ‘♣’ indicates adopting Sigmoid and Softmax activation function, respectively. The best performance in each column is boldfaced.

| Setting | $L_{Conv}$ | $G_{Trm}$ | Toys | Beauty | Sports |
|---------|------------|-----------|------|--------|--------|
| (1)     | ♦          | ♦         | 0.3343 | 0.3447 | 0.3226 |
| (2)     | ♣          | ♦         | 0.3357 | 0.3433 | 0.3316 |
| (3)     | ♦          | ♣         | 0.3355 | 0.3525 | 0.3255 |
| (4)     | ♦          | ♣         | 0.3511 | 0.3554 | 0.3444 |

fix the global and local trade-off to a constant value without userspecific learnable adaptability, the recommendation performance drops drastically. The best fixed AdaMCT using 80% global and 20% local modules without personalized adaptability is only slightly better than BERT4Rec but worse than the self-adaptive AdaMCT. Table 3 also shows that the proposed AdaMCT that adaptively exploits the mixing importance of CNNs and Transformer layer by layer can improve the overall representation learning effectively.

4.3 Ablation Study (RQ2)
To understand the importance of different modules in our model, we conduct a detailed ablation study. We analyze the importance of self-adaptive local (CNNs) and global (Transformer) awareness in Table 3, analyze the functions of various computational components in Figure 3, and analyze the effectiveness of squeeze-excitation attention (SEAtt) in Table 5.

As can be seen in Figure 3, when we remove any of the proposed components of AdaMCT, the performance drops by a significant margin, which demonstrates the effectiveness of each designed component. Specifically, the ablation results reveal that removing the global attention module (Transformer) $G_{Trm}$ and local convolutional module (CNNs) $L_{Conv}$ will significantly drop the model performance across all three datasets. It confirms our intuition that local and global features are important for sequential recommendation tasks. Moreover, adaptive mixture unit $A_{Mix}$ plays a vital role in mixing the local (CNNs) and global (Transformer) features. Removing two squeeze-excitation attention (SEAtt) also leads to performance decline. However, different datasets have shown different sensitivity patterns. It seems that the simpler Beauty dataset is easier to be learned the importance of each item compared to the other two datasets.

We discuss the effectiveness of the squeeze-excitation attention (Sigmoid) in Table 5. The proposed SEAtt can improve the original Softmax approach by approximately 5.03%, 2.73%, and 6.29% over three datasets respectively, in terms of NDCG@10. Besides, we observe that Sigmoid is more beneficial to the local branches (CNNs), and Sigmoid is more favorable for the global branches (Transformer). In addition, the performance of Sigmoid on the local branch still outperforms the performance of Softmax on the global branch, and the hybrid of Softmax and Sigmoid is better than simply adopting the Softmax on both branches except the Beauty dataset. These demonstrate that considering multiple relevant items of equal importance simultaneously on user historical behavior is conducive to capturing user interest preference accurately.

4.4 Hyperparameter Study (RQ3)
In this section, we conduct a stability study to discuss the effect of a variety of hyperparameters on model performance in Table 4. We also visualize the learning procedure of local-global coefficients to get the gist of how AdaMCT adaptively mixes the feature of CNNs and Transformers, as can be seen in Figure 4(d).

Specifically, we analyze the effect of the kernel size ($k$), reduction ratio ($r$), hidden activation ($Φ$), head number ($h$), and hidden dimension ($d_{model}$) in Table 4. It can be seen that on all three datasets, the performance of AdaMCT improves by a narrow margin with changes in dimensionality. This shows that our model is generally insensitive to dimensionality. In particular, the smaller size of the hidden dimension ($d_{model}$) shows a better performance, indicating that for simpler datasets (e.g., Beauty) the hidden dimensionality does not need to be too large. From the results, we can tell that AdaMCT is overall stable.

One of the novelties in the proposed AdaMCT model lies in the fusion of local (CNNs) and global (Transformer) information through a self-adaptive approach. To validate our intuition, we visualize the learning procedure of coefficients of the local and global branches for two randomly chosen users (denoted as user1 ($P_1$) and user2 ($P_2$)), two layers (denoted as $L_1$ and $L_2$) and the $α$ and $β$ of the last layer with average scores, as shown in Figure 4(d). We observe that AdaMCT updates the coefficients through
In order to understand whether the performance improvement is attributable to the model architecture design or the escalation in the quantity of parameters, we conduct an analysis on AdaMCT’s complexity and execution efficacy, as delineated in Table 6. Comparing with CNN-based and Transformer-based baselines, we have the following observations: (1) AdaMCT has relatively fewer parameters (76%-93% less). This is reasonable because our Transformer removes Feed-forward Network (FFN) and becomes more lightweight. (2) AdaMCT has better efficiency in both training (86% less time) and inference (83% less time) compared with Caser. Meanwhile, AdaMCT achieves comparable training/inference time and a much smaller memory footprint (66% less) and FLOPs (93% less) with SASRec and CORE. Moreover, our model has the minimum FLOPs compared with all baselines which further enhances the feasibility of practical deployment, especially for low-resource constraints devices. The results indicate that AdaMCT is capable of achieving a good balance between computational cost and recommendation accuracy.

### 4.6 Visualization (RQ5)

To provide insights into the AdaMCT’s interpretability, we visualize the item correlation matrices based on feature representation of $\mathcal{L}_{\text{Conv}}$, $\mathcal{G}_{\text{Trm}}$, and $\mathcal{A}_{\text{Mix}}$ output in the last layer of AdaMCT, using test samples from the Beauty dataset, as shown in Figure 4(a)(b)(c). Based on our observations, we find that different components have different item correlation patterns. $\mathcal{L}_{\text{Conv}}$ tends to learn local item dependencies which concentrate more on recent items (red-orange area with higher correlation scores), while $\mathcal{G}_{\text{Trm}}$ prefers to learn items dependencies globally, e.g., the 3rd item has the higher correlation with other items, as can be seen in Figure 4(a) and (b) green dash rectangular box, respectively.

### 5 CONCLUSION

In this paper, we design a novel Adaptive Mixture of CNN-Transformer (AdaMCT) that injects locality inductive bias into Transformer by combining its global attention mechanism with a local convolutional filter, and adaptively determines the mixing importance on items dependencies. Extensive experiments on three public real-world datasets demonstrate the effectiveness and efficiency of our proposal. For future studies, we will explore how to design more powerful adaptive mixture units in-depth and incorporate item-related (e.g., price, date of production, producer) or behavior-related (e.g., purchase, rate) side information into AdaMCT.

### 6 ACKNOWLEDGMENTS

This work was partially supported by the National Natural Science Foundation of China (No. 62276099). We appreciate the NVIDIA A100 SXM4 80GB GPUs support of Upstage.
[48] Xiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. 565–573.

[49] Xiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. 565–573.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems. 5998–6008.

[51] Junling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item Recommendation with Sequential Hypergraphs. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 1101–1110.

[52] Yaqi Wang, Chaozhuo Li, Zheng Liu, Mingzheng Li, Jiliang Tang, Xing Xie, Lei Chen, and Philip S Yu. 2022. An adaptive graph pre-training framework for localized collaborative filtering. ACM Transactions on Information Systems 41, 2 (2022), 1–27.

[53] Liwei Wu, Shuaqing Li, Cho-Jui Hsieh, and James Sharpnack. 2020. SSE-PT. Sequential recommendation via personalized transformer. In Fourteenth ACM Conference on Recommender Systems. 328–337.

[54] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 346–353.

[55] Ruobing Xie, Yalong Wang, Rui Wang, Yufan Lu, Yuanhang Zou, Feng Xia, and Leyu Lin. 2022. Long short-term temporal meta-learning in online recommendation. In Proceedings of the Fiftteenth ACM International Conference on Web Search and Data Mining. 1168–1176.

[56] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuizhen Zhuang, Junhua Fang, and Xiaofang Zhou. 2019. Graph Contextualized Self-Attention Network for Session-based Recommendation. In IJCAI, Vol. 19. 3940–3946.

[57] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Jiajie Xu, Victor S Sheng, Sheng Zhang, Zhiming Cui, Xiaofang Zhou, and Hui Xiong. 2019. Recurrent convolutional neural network for sequential recommendation. In The World Wide Web Conference. 3398–3404.

[58] Yong Xu, Jiahuai Chen, Chao Huang, Bo Zhang, Hao Xing, Peng Dai, and Liefeng Bo. 2020. Joint Modeling of Local and Global Behavior Dynamics for Session-based Recommendation. In FCAI 2020 IOS Press. 545–552.

[59] Meng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In Proceedings of the 39th ACM SIGIR conference on Research and Development in Information Retrieval. 729–732.