Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.057224

ARTICLE

MixerKT: A Knowledge Tracing Model Based on Pure MLP Architecture

Jun Wang[1,2], Mingjie Wang[1,2], Zijie Li[1,2], Ken Chen[1,2], Jiatian Mei[1,2] and Shu Zhang[1,2,*]

[1]Key Laboratory of Education Informatization for Nationalities, Ministry of Education, Yunnan Normal University, Kunming, 650000, China

[2]Yunnan Key Laboratory of Smart Education, Yunnan Normal University, Kunming, 650000, China

*Corresponding Author: Shu Zhang. Email: zhangshu@ynnu.edu.cn

Received: 11 August 2024 Accepted: 07 October 2024 Published: 03 January 2025

ABSTRACT

In the field of intelligent education, the integration of artificial intelligence, especially deep learning technologies, has garnered significant attention. Knowledge tracing (KT) plays a pivotal role in this field by predicting students' future performance through the analysis of historical interaction data, thereby assisting educators in evaluating knowledge mastery and tailoring instructional strategies. Traditional knowledge tracing methods, largely based on Recurrent Neural Networks (RNNs) and Transformer models, primarily focus on capturing long-term interaction patterns in sequential data. However, these models may neglect crucial short-term dynamics and other relevant features. This paper introduces a novel approach to knowledge tracing by leveraging a pure Multilayer Perceptron (MLP) architecture. We propose MixerKT, a knowledge tracing model based on the HyperMixer framework, which uniquely integrates global and local Mixer feature extractors. This architecture enables more effective extraction of both long-term interaction trends and recent learning behaviors, addressing limitations in current models that may overlook these key aspects. Empirical evaluations on two widely-used datasets, ASSISTments2009 and Algebra2005, demonstrate that MixerKT consistently outperforms several state-of-the-art models, including DKT, SAKT, and Separated Self-Attentive Neural Knowledge Tracing (SAINT). Specifically, MixerKT achieves higher prediction accuracy, highlighting its effectiveness in capturing the nuances of learners' knowledge states. These results indicate that our model provides a more comprehensive representation of student learning patterns, enhancing the ability to predict future performance with greater precision.

KEYWORDS

Knowledge tracing; multilayer perceptron; channel mixer; sequence mixer

Nomenclature

MixerKT	Knowledge tracing model based on HyperMixer
KT	Knowledge tracing

1 Introduction

With the ongoing advancement of artificial intelligence, the topic of smart education has garnered significant attention among researchers [1]. The rise of deep learning technologies presents new avenues

for application in the education sector, particularly making the integration of multilayer perceptron neural networks into educational intelligence algorithms a burgeoning area of interest. Central to smart education research is the concept of knowledge tracing, which seeks to forecast students' future performance by analyzing their historical interactions, primarily focusing on responses to previous questions. Fig. 1 illustrates the knowledge tracking task. This approach empowers educators and digital learning platforms to understand learners' current knowledge levels, identify misunderstandings, and tailor future teaching methods or learning trajectories accordingly. Knowledge tracing has found applications in online education platforms like Duolingo and Shanbei English [2].

Figure 1: Knowledge tracing task

From the perspective of sequence modeling, knowledge tracing techniques are generally categorized into two groups: those based on Recurrent Neural Networks (RNN) and those leveraging Transformers [3]. These methodologies adopt a common framework for sequence tasks, encapsulating historical data through an encoder and employing the resultant hidden states for predictions at designated time intervals. This makes the adaptation of strategies from areas such as recommendation systems and natural language processing to knowledge tracing not only viable but also highly effective.

Recently, researchers have developed a new pure multilayer perceptron (MLP) architecture known as MLP-Mixer [4]. This architecture, with its unique feature mixing layers, achieves impressive performance in various tasks, especially in handling sequence data, demonstrating its strong potential. A key characteristic of MLP-Mixer is its ability to mix feature information globally, enabling the model to capture long-range dependencies while maintaining detailed processing of local features, which is crucial for sequence tasks. Following this, Li et al. [5] successfully applied the MLP-Mixer architecture to recommendation system tasks, showcasing its effectiveness in sequence recommendation. Subsequently, Mai et al. [6] applied MLP-Mixer in the field of natural language processing, proving its potential in processing language sequences.

Given the architecture's excellent performance in related fields, we believe that Mixer has significant potential in sequence modeling and may outperform LSTM or Transformer in some scenarios. To further enrich the modeling techniques used for knowledge tracing tasks, this paper attempts to introduce Mixer into the knowledge tracing field to discuss whether the technology can achieve similar performance. The applications of pure MLP solutions in the current research that we discussed in Related Work and the advantages of Mixer in various fields and Mixer introduced the advantages of using MLP and Mixer to solve the field of knowledge tracking. Subsequently, we introduced MixerKT in the Model section, introduced Embedding Layer and the core Mixer Layer, and used the binary cross-entropy loss for training. We conducted experiments on the Assistments2009 and Algebra2005 data sets, and chose the more classic models in the field of knowledge tracking and the recent new models as the baseline for comparative experiments. We have conducted ablation experiments

to verify the contribution of each key module to MixerKT. We also experimented with several key super parameters of MixerKT. Our specific contributions are: (1) proposing a new knowledge tracing model based on the Mixer architecture, known as MixerKT, which to our knowledge, is the first application of the Mixer architecture in the field of knowledge tracing; (2) introducing a method combining global Mixer feature extractors and local Mixer feature extractors. The global Mixer models the entire data sequence. We recognize the importance of short-term performance in evaluating a learner's knowledge. Therefore, we use a local Mixer to focus on recent learner features; (3) demonstrating through empirical evidence that MixerKT achieves competitive performance on datasets like ASSISTments2009 compared to advanced baseline models. Distinctively, MixerKT surpasses existing MLP-based models, such as MLP-Mixer and MLP4Rec, by integrating global and local Mixer feature extractors, enabling a more nuanced understanding of learners' interaction patterns and recent learning behaviors.

2 Related Work

2.1 Knowledge Tracing

The goal of knowledge tracing is to predict students' future performance in interactions based on their historical interaction records. Currently, the field has established a stable paradigm where the model input is the student's interaction record, and the output is the student's performance at a future moment. Existing datasets record students' responses to a sequence of exercises over a period, either through automatic system collection or manual gathering, making this task highly related to other sequence modeling tasks. The task definition of knowledge tracing can be formalized as follows: for a given learner interaction sequence, the interactions produced by the learner are represented as a series of triplets, $s_t = [(c_1,\ q_1,\ r_1),\ (c_2,\ q_2,\ r_2), \ldots,\ (c_t,\ q_t,\ r_t)]$. Here, q_t represents the exercise answered by the learner at time t, c_t represents the concept of the knowledge point corresponding to the exercise at time t, and r_t indicates whether the learner answered the exercise correctly at time t, with a value of 1 if correct. The task of a knowledge tracing model is to predict the learner's subsequent interaction performance based on the sequence s_t, specifically predicting the performance of the subsequent interaction exercise q_{t+1}, i.e., given s_t and q_{t+1} to predict r_{t+1}.

From the perspective of capturing learner sequence features, knowledge tracing can be broadly divided into methods based on Recurrent Neural Networks (RNN) and those based on Transformer architecture. RNN-based methods appeared earlier, with DKT [7] being the first model to implement knowledge tracing tasks using LSTM/RNN. Subsequent improvements by researchers include the DKT+ model [8], which optimizes the model training objective using a loss function to make DKT comply with basic cognitive principles, the DKT-forget model [9] that integrates the forgetting process, and the AT-DKT [10] through multi-objective optimization. Recent models like LPKT [11] and QIKT [12] are also based on the RNN architecture. After achieving outstanding results in natural language processing, researchers also introduced the Transformer architecture to knowledge tracing. SAKT [13] uses a self-attention mechanism, similar in structure to the Transformer. SAINT: Separated Self-Attentive Neural Knowledge Tracing [14] was the first to use the original Transformer architecture to build a knowledge tracing model. AKT [15] achieved exceptional performance with Rasch-based embeddings and a custom attention mechanism. Additionally, models utilizing other technologies or training methods have also at least adopted the RNN and Transformer architectures, such as GKT [16] using Graph Neural Networks, BiCLKT [17], and CL4KT [18] based on contrastive learning.

2.2 Mixer

In the field of computer science, compared to models based on the Transformer architecture, MLP (Multilayer Perceptron) architectures generally exhibit lower time and space complexity. Recent research has shown that MLP architectures, with their straightforward design, can achieve performance comparable to state-of-the-art technologies.

The introduction of MLP-Mixer has provided a purely MLP-based methodology for computer vision tasks. This architecture integrates two MLPs, one for feature mixing and another for token mixing, effectively simulating spatial and feature-level interactions within the data. This strategy allows the model to capture the intrinsic structure of the data without relying on complex mechanisms specific to certain data types. In the research by Li and others, MLP4Rec [5] was proposed, applying the MLP-Mixer [6] architecture to the domain of sequence recommendation. It addresses the limitations of self-attention models (such as quadratic computational complexity and dependency on positional embeddings) and proposes a tri-directional fusion scheme to capture correlations across sequences, channels, and features, while maintaining linear computational complexity and fewer model parameters. To overcome the limitations of MLP4Rec in handling users' long/short-term interests, Li et al. further introduced AutoMLP [19], incorporating an automated and adaptive algorithm to determine the length of short-term interests. This approach not only maintains linear computational complexity but also demonstrates comparable performance to advanced methods in sequence recommendation tasks. In the study of HyperMixer, the MLP architecture was applied to the field of natural language processing. This model, by dynamically forming token-mixing MLPs using hypernetworks, effectively addresses the limitations of MLP-Mixer in natural language understanding. HyperMixer not only rivals traditional Transformer models in performance but also significantly reduces costs in terms of processing time, training data, and hyperparameter tuning.

Overall, from the initial practice of MLP-Mixer to the application of MLP4Rec and AutoMLP in sequence recommendation, and the innovation of HyperMixer in the NLP domain, the MLP architecture has undergone continuous development and refinement. It has brought new developmental potential and application possibilities to multiple areas of machine learning.

3 Model

We propose a pure MLP knowledge tracing model based on the Hyper-Mixer architecture, named MixerKT. This model comprises an embedding layer, Mixer layers, and a prediction layer, with its structure illustrated in Fig. 2. The embedding layer is used to achieve vector representations of knowledge points and exercises. The Mixer layers are responsible for capturing the temporal and feature information of the knowledge point sequences. The prediction layer extracts the necessary sequence information and exercise details for making predictions and outputs the results. To train the model, this section will also introduce the loss function and training process used for MixerKT. The choice of a Multilayer Perceptron (MLP) architecture over traditional Recurrent Neural Networks (RNN) or Transformer architectures is rooted in the unique requirements of knowledge tracing. MLPs provide a straightforward and efficient means of capturing the non-sequential relationships between learning interactions, allowing for faster training and inference. Unlike RNN, which can struggle with long-term dependencies, or Transformers, which may introduce unnecessary complexity for the specific task of knowledge tracing, the MLP architecture effectively balances model simplicity with the capacity to learn from historical interaction data. This enables MixerKT to focus on both local and global features without the drawbacks of sequential processing limitations.

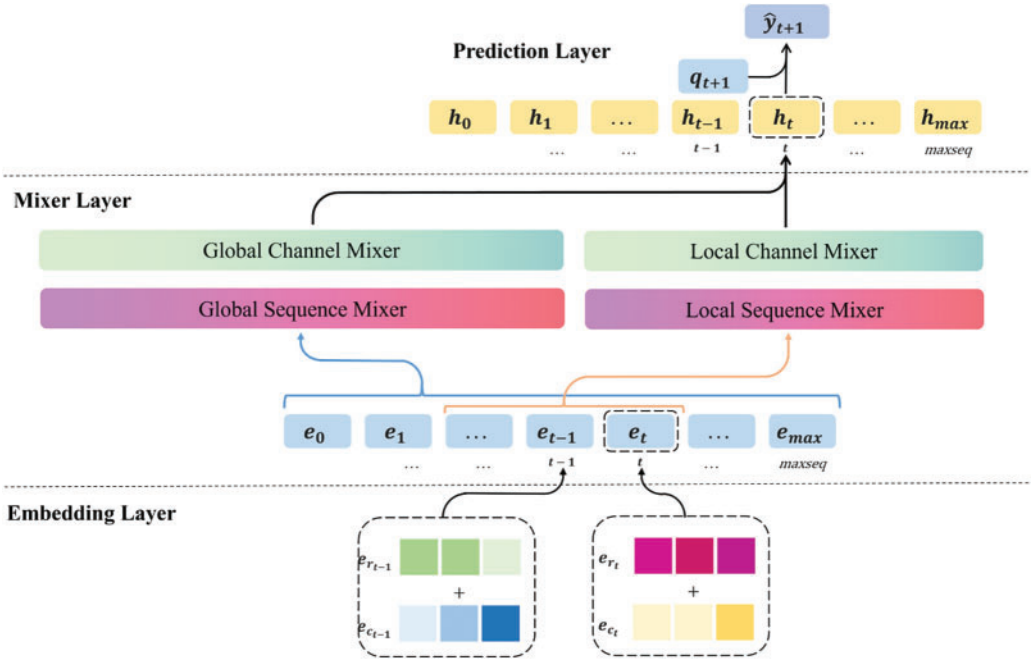

Figure 2: MixerKT model structure

3.1 Embedding Layer

In accordance with the task definition of knowledge tracing, MixerKT implements vector representations of knowledge points and exercises in the embedding layer, a process similar to other knowledge tracing works. Specifically, we embed each concept within a set of N knowledge points, $C = \{c_1, c_2, \ldots, c_N\}$, with an embedding dimension of d. This results in a knowledge point embedding matrix E_C in $R^{N\times d}$. The embedding approach for exercises is consistent with that of knowledge points, using the same embedding dimension d, leading to an embedding matrix E_Q in $R^{M\times d}$, where M is the number of exercises. Since a learner's response can only be correct or incorrect, it is necessary to embed these response scenarios, denoted as: $E_R in R^{2\times d}$. As illustrated, for the interactions already generated by the learner, the final representation of a student's interaction at time t is processed based on their response to the knowledge point, such that the final representation at time t, e_t is the sum of the embeddings of the knowledge point and the response at that time, $e_t = e_{c_t} + e_{r_t}$.

3.2 Mixer Layer

In the Mixer layer, the focus is on capturing global sequence features and local features. In past knowledge tracing work, models typically use LSTM or Transformer to model sequences, employing the hidden states outputted by LSTM or Transformer as representations of a student's knowledge state. Similarly, the function of Mixer is to capture the hidden states of the sequence, with the notable difference that Mixer's capture does not rely on the results of the previous time step. Furthermore, since MixerKT is based on a pure MLP architecture, it also benefits from lower computational overhead. The Mixer layer is composed of two independent Mixer Blocks: one known as the global Mixer feature extractor and the other as the local Mixer feature extractor. Each Block contains a Sequence Mixer and a Channel Mixer, following their respective operational mechanisms, as shown in Fig. 3.

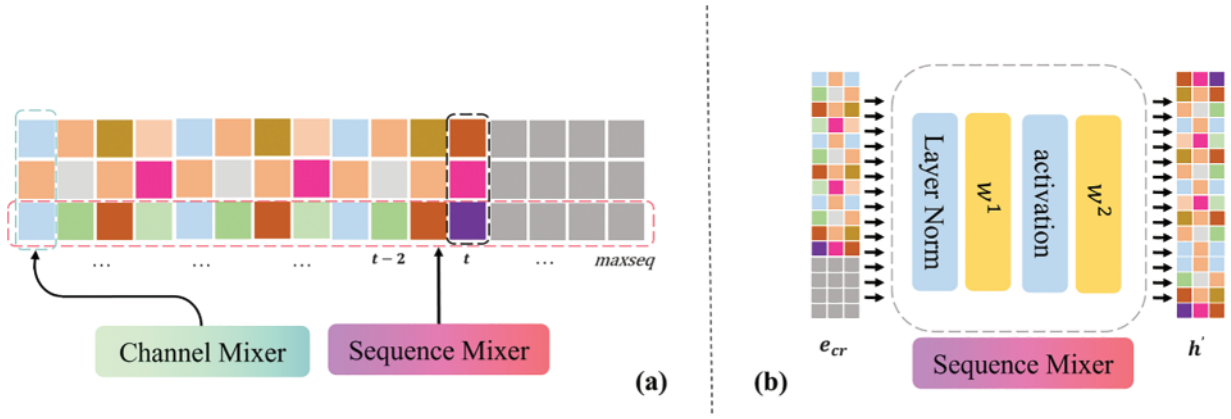

Figure 3: The structure of mixer. (a) Illustrates sequence and channel mixing operations in the mixer layer. (b) Illustrates nonlinear activation function (GELU) in the mixer layer

The Mixer layer is implemented using a Sequence Mixer and a Channel Mixer, where the Sequence Mixer's purpose is to mix information between different tokens of the sequence, and the Channel Mixer aims to mix information across different feature channels, as illustrated in Fig. 3a. Assuming a sequence representation X in $R^{N\times d}$, where N is the sequence length and d is the feature dimension, the forms of the Sequence Mixer and Channel Mixer are as shown in Eqs. (1) and (2).

$$\text{Sequence Mixer } (\mathrm{X}) = \text{Activation } (X \cdot W_1 + b_1) \tag{1}$$

$$\text{Channel Mixer } (X) = \textit{Activation } (X^T \cdot W_2 + b_2) \tag{2}$$

In the Mixer layer, the weight matrices for the Sequence Mixer and Channel Mixer are denoted as W_1 and W_2, respectively, with bias vectors b_1 and b_2. The nonlinear activation function used is GELU, as depicted in Fig. 3b.

The weight matrices W_1 and W_2 are generated by parameterized functions $h_1, h_2 \colon R^{N\times d} \to R^{N\times d'}$, where we choose to independently generate the rows of each weight matrix from each token using another MLP. The definition of the hypernetwork function is given in Eq. (3), with p_j in R^d being the vector that includes additional information through position encoding.

$$h_i(x) = \begin{pmatrix} MLP^{W_i}(x_1 + p_1) \\ \vdots \\ MLP^{W_i}(x_N + p_N) \end{pmatrix} \in \mathbb{R}^{N\times d'} \tag{3}$$

The global Mixer feature extractor is designed to model the entire sequence, thus considering the sequence features at time t from the perspective of the entire sequence. The local Mixer feature extractor aims to capture the features from the most recent k time steps before t, motivated by the idea that a student's recent interactions could be highly significant. Here, k is a hyperparameter of the model. Assuming a maximum sequence length of max, the input to the global Mixer feature extractor is $S^G = [e_0, e_1, \ldots, e_{t-1}, e_t, \ldots, e_{max-1}]$ in $R^{max\times d}$, and for the local Mixer feature extractor, the input is $S^L = [e_{t-k+1}, \ldots, e_{t-1}, e_t]$ *in* $R^{k\times d}$, where e_t represents the result of the knowledge point completed by the student at time t. We utilize seq to facilitate interactions between sequences and Channel for interactions within features, enabling the generated sequence features to possess temporal information while also integrating internal feature information, as shown in Eqs. (4) and (5).

$$H^G = \text{Channel Mixer (Sequence Mixer } (S^G)) \tag{4}$$

$$H^L = \text{Channel Mixer (Sequence Mixer } (S^L)) \tag{5}$$

The fused results are $H^G = [h_0^G, \ldots, h_{t-1}^G, h_t^G, \ldots, h_{max}^G]$ in $R^{max \times d}$ and $H^L = [h_{t-k+1}^L, \ldots, h_{t-1}^L, h_t^L]$ in $R^{k \times d}$. It is important to note that when predicting the student's performance at time t+1, all moments in the $[t+1, max]$ interval are masked to prevent label leakage. Finally, the output results of the two feature extractors at time t are concatenated to form h_t, as illustrated in Eq. (6).

$$h_t = h_t^G \oplus h_t^L \in \mathbb{R}^{2d} \tag{6}$$

The sequence feature at time t, represented as h_t, is then used to predict the student's performance at time $t+1$.

3.3 Prediction Layer

The prediction layer is utilized to achieve the prediction of results. If predicting the student's performance at time $t+1$, denoted as $\hat{y}_{t+1}$, the input to this layer is the output result h_t from the Mixer layer and the exercise q_{t+1} that is to be predicted at time $t+1$. The prediction layer computes the prediction result using an MLP layer and a Sigmoid function, as shown in Eq. (7).

$$\hat{y} = \sigma\ (W\ [h_t \oplus q_{t+1}] + b) \tag{7}$$

Here, σ denotes the sigmoid function, and W represents the parameters of the fully connected layer. This layer operates to reduce dimensions through a neural network while adjusting the output to fall within the [0, 1] interval.

3.4 Loss and Training

The model employs a binary cross-entropy loss function to compute scores for each sequence at every time step, with the loss for a single sequence represented as shown in Eq. (8).

$$L = -\sum_{t=1}^{T} (y_t\ log\ \widehat{y_t} + (1 - y_t)\ log\ (1 - \widehat{y_t}) \tag{8}$$

In this equation, y_t represents the true label of the student's interaction. The model is optimized using the Adam optimizer, with the training objective being the minimization of loss.

4 Experiment

We conducted experiments on the ASSISTment2009 [20] and Algebra2005 [21] datasets using the pyKT toolkit [22], comparing performance with baseline models such as DKT, SAKT, and SAINT under a five-fold cross-validation setting through both Question Level and KC (Knowledge Component) Level approaches. Additionally, ablation studies and hyperparameter experiments were conducted on the ASSISTment2009 dataset. All experiments were completed on a uniform platform: the computational core was powered by an Intel Xeon E-2288G octa-core processor, and the GPU core was supported by an NVIDIA A100 PCIe (80 GB HBM2e) to facilitate high-load deep learning tasks. During training, an early stopping strategy was employed with training epochs set to 50 and batch sizes chosen from {32, 64}. The learning rates were set within {0.001, 0.005}, and the embedding dimension d for exercises was chosen from {64, 128, 256} to prevent overfitting, a dropout of 0.2 was set.

4.1 Dataset

The ASSISTment2009 dataset (abbreviated as AS2009) is a collection designed for student cognitive diagnostics and knowledge tracing research. After processing by pyKT, the dataset comprises a total of 337,415 interaction records, 4661 sequences, 17,737 problems, and 123 distinct knowledge points. It contains student exercise data collected during the 2009 to 2010 academic year through the ASSISTments online teaching platform. This dataset includes information about students' interaction records, sequential data, the problems themselves, and related knowledge points. The AS2009 dataset provides real-world data for evaluating student model construction and knowledge tracing algorithms, making it widely used in performance assessment and comparative analysis of various knowledge tracing methods.

The Algebra2005 dataset (abbreviated as AL2005), originating from the KDD Cup 2010 educational data mining challenge, encompasses responses of 13 to 14-year-old students to algebra problems. The processed Algebra2005 dataset includes 884,098 student interaction records, covering 4712 learning sequences and 173,113 problems, as well as 112 different knowledge points.

4.2 Comparative Experiment

The MixerKT model was compared with several benchmark models in experimental studies. These benchmarks include the deep knowledge tracing model DKT and its variant DKT+, as well as the attention-mechanism-incorporating SAKT model. Each of these models underwent performance evaluations at both Question Level and KC Level on the AS2009 and AL2005 datasets. The MixerKT model employs a pure Multilayer Perceptron (MLP) architecture, distinctly contrasting with the common knowledge tracing models based on Recurrent Neural Networks (RNN) or attention mechanisms.

Table 1 shows the performance comparison of the MixerKT model with baseline models on the AS2009 and AL2005 datasets at both Question Level (All-in-One) and KC Level (All-in-One). It was found that the MixerKT model outperforms other baseline models in both Question Level and KC Level performance on these two datasets.

Table 1: Comparative experimental results of the MixerKT model with baseline models at Question Level and KC Level on the AS2009 and AL2005 datasets. ΔDKT represents a value that is superior to the suboptimal alternative

Model	Question level (all-in-one)		KC level (all-in-one)	
	AS2009	AL2005	AS2009	AL2005
DKT [7]	0.7541	0.8149	0.7419	0.8146
DKT+ [8]	0.7547	0.8156	0.7424	0.8144
DKT-F [9]	–	0.8147	–	0.8163
KQN [23]	0.7477	0.8027	0.7361	0.8005
DKVMN	0.7473	0.8054	0.7330	0.7891
ATKT [24]	0.7470	0.7995	0.7337	0.7964
GKT [16]	0.7424	0.8110	0.7227	0.8025
SAKT [13]	0.7246	0.7880	0.7085	0.7682
SAINT [14]	0.6958	0.7775	0.6865	0.6662

(Continued)

Table 1 (continued)

Model	Question level (all-in-one)		KC level (all-in-one)	
	AS2009	AL2005	AS2009	AL2005
AT-DKT [10]	0.7421	0.7799	0.7498	0.7911
MixerKT	0.7646	0.8266	0.7581	0.8193
ΔDKT	0.0105	0.0117	0.0162	0.0047

Specifically, at the Question Level, the accuracy of the MixerKT model on the AS2009 dataset improved by 1.05% compared to the DKT model and by 4% compared to the SAKT model. On the AL2005 dataset, the improvement was 1.17% over the DKT model. At the KC Level, the MixerKT model also showed significant improvements. On the AS2009 dataset, its accuracy was 1.62% higher than the DKT model and 4.96% higher than the SAKT model. On the AL2005 dataset, it outperformed the DKT model by 0.47% and the SAKT model by 5.11%. These results indicate that even without adopting complex sequence model structures, a pure MLP architecture can effectively capture students' knowledge states and learning progress.

4.3 Ablation Study

In the ablation study, we aimed to quantitatively analyze the contributions of various components of the MixerKT model. The experiment involved removing specific components from the base architecture of MixerKT and evaluating their impact on the overall performance of the model. The performance metrics considered included AUC (Area Under the Receiver Operating Characteristic Curve) and ACC (Accuracy), both of which are widely used in the machine learning field to assess the performance of classifiers. The experimental setup comprised four configurations: Configuration I utilized the basic Mixer module, testing the model's performance without the HyperMixer component; Configuration II removed the Local Mixer Block to assess its impact on local feature processing; Configuration III further removed both the Local Mixer Block and the HyperMixer module, helping to understand the combined effect of these two components; Configuration IV represented the complete MixerKT model, combining all components as a performance benchmark.

The results of the ablation study, as shown in Table 2, indicate that the complete MixerKT model (Configuration IV) outperformed other configurations in both AUC and ACC metrics. Compared to Configuration I, replacing HyperMixer with Mixer resulted in a decline of 0.33%, 0.34%, 0.31%, and 0.3% in AUC and ACC at both Question Level and KC Level, respectively. This finding suggests that while the contribution of HyperMixer to accuracy is small, it still has a positive effect, playing an active role in local context modeling and feature extraction. In Configuration II, where the Local Mixer Block was removed, there was a more significant performance drop of 0.42% in AUC and 0.49% in ACC at the Question Level, highlighting the role of the Local Mixer Block in the overall architecture. It not only enhanced the model's sensitivity to local context but also improved the capture of fine-grained changes in students' knowledge states. Configuration III further demonstrated the performance decline when both key components were removed, with a decrease of 0.71% in AUC and a drop to 0.75% in ACC at the Question Level. The significance of this combined effect indicates that the Local Mixer Block and HyperMixer not only function independently within the model but their interactive effects are crucial for enhancing model performance. The HyperMixer may engage in

feature interactions at a global level, while the Local Mixer Block enhances the representation of local features. Together, they provide the model with richer and more robust learning capabilities.

Table 2: Ablation study results of MixerKT

Model	Question level (all-in-one)		KC level (all-in-one)	
	AUC	ACC	AUC	ACC
I w/o HyperMixer (using mixer)	0.7613	0.7211	0.7549	0.7176
II w/o Local Mixer Block	0.7604	0.7197	0.7536	0.7158
III w/o Local Mixer Block and HyperMixer	0.7575	0.7171	0.7506	0.7136
IV MixerKT	0.7646	0.7246	0.7580	0.7206

The superior performance of the MixerKT model is attributed to the synergistic operation of its internal components, with both the Local Mixer Block, enhancing local features, and the HyperMixer, facilitating the integration of global information, playing indispensable roles in the model's effectiveness and accuracy.

4.4 Hyperparameter Experiment

We conducted hyperparameter tuning experiments focusing on the most critical hyperparameters: the embedding dimension d and the number of recent learning interactions k captured by the short sequence. The goal was to assess the impact of different hyperparameter values on the performance of the MixerKT model.

Initially, experiments were conducted on varying embedding dimensions d, with values set at 64, 128, and 256. These experiments were evaluated at the Question Level and KC Level on the AS2009 dataset, using AUC and ACC as performance metrics.

Table 3: Performance of MixerKT at different embedding dimensions

Model	Question level (all-in-one)		KC level (all-in-one)	
	AUC	ACC	AUC	ACC
$d = 64$	0.7609	0.7204	0.7546	0.7170
$d = 128$	0.7581	0.7181	0.7516	0.7143
$d = 256$	0.7630	0.7215	0.7569	0.7184

The results, as shown in Table 3, indicate significant differences in model performance across various embedding dimensions. Specifically, at the Question Level, with $d = 64$, the model achieved an AUC of 0.7609 and an ACC of 0.7204. Increasing the dimension to $d = 128$ resulted in a slight decrease in AUC to 0.7581 and ACC to 0.7181, suggesting that a lower dimension might be sufficient to capture patterns in the dataset. Further increasing the dimension to $d = 256$ led to an increase in AUC to 0.7630 and ACC to 0.7215, indicating that higher dimensions provide a richer representation space, potentially improving model performance. At the KC Level, with $d = 64$, the model's AUC and ACC were 0.7546 and 0.7170, respectively. With $d = 128$, AUC slightly decreased to 0.7516, but ACC

marginally increased to 0.7143, implying that an increase in dimension at this level did not significantly enhance performance. With $d = 256$, the model's performance improved again, with AUC and ACC reaching 0.7569 and 0.7184, respectively, reiterating the positive impact of higher dimensions on model performance.

The results reveal a correlation between model dimension and performance, showing an upward trend in AUC and ACC as the dimension increases, particularly when increasing from 128 to 256. However, the performance improvement was not as pronounced when increasing from 64 to 128, indicating a possible dimensional threshold for the given dataset and task. Beyond this threshold, the model might more effectively learn complex patterns in the data.

We then evaluated the performance of the Local Mixer feature extractor with different k values, where k represents the number of recent features captured by the model before time point t. This was achieved by observing changes in window AUC values under different sequence lengths.

From Fig. 4, it is observed that as k increased from 1 to 7, the window AUC rose from 0.7493 to 0.7569, indicating that expanding the context window significantly enhances model performance. A smaller k value may limit the amount of information captured by the model. However, as this window expands, the model gains access to more historical interaction data, leading to better understanding and prediction at the current moment. In the k range of 7 to 10, the window AUC showed a decreasing trend, dropping from 0.7569 to 0.7530. The model might have already captured sufficient information for effective prediction at $k = 7$, and additional information did not lead to performance improvement. Between $k = 10$ and $k = 15$, the window AUC displayed stability but with a slight downward trend. Specifically, the AUC was 0.7530 at $k = 10$ and slightly decreased to 0.7520 at $k = 15$. As k increased, the model began to consider more distant historical interactions, which might be less relevant to the current prediction, resulting in a performance decline.

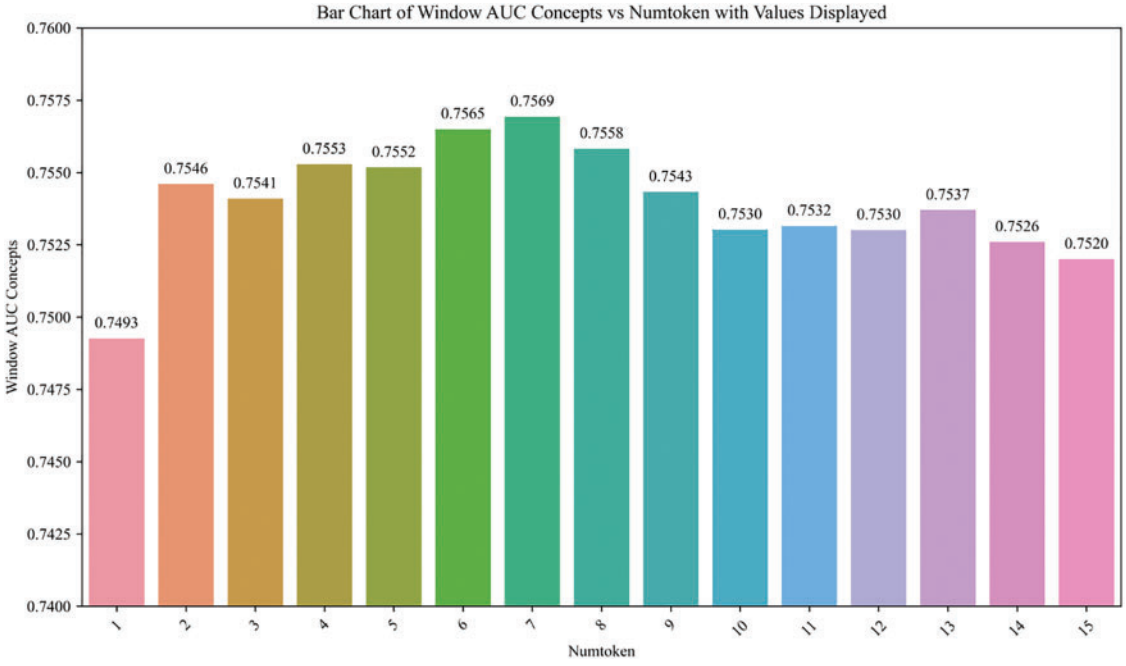

Figure 4: Experimental results of window AUC values at different sequence lengths k

The results suggest that there exists an optimal k value, or window size, where the model achieves peak performance. Beyond this optimal point, performance decline may be due to the model overfitting or processing too much less relevant information, reducing its focus on the most recent and significant interactions.

5 Conclusion

In this study, we successfully introduced the MLP-Mixer architecture into the field of knowledge tracing and developed the innovative MixerKT model. This model integrates both global and local Mixer feature extractors, effectively capturing students' overall and recent interaction features to enhance the accuracy of predicting their future performance. Experimental results demonstrate that MixerKT outperforms existing knowledge tracing models based on recurrent neural networks and Transformer architectures on standard datasets such as Assistment2009 and Algebra2005. This breakthrough not only confirms the applicability of the Mixer architecture in the domain of education but also opens new technological pathways and research perspectives for the development of future intelligent educational systems. By deeply understanding students' learning patterns and knowledge states, MixerKT paves the way for improving educational quality and efficiency through intelligent technology.

Future research directions for MixerKT could involve exploring its application in other domains, such as personalized learning systems or adaptive assessment environments, where understanding learner interactions is crucial. Additionally, efforts to further optimize the model's computational efficiency could enhance its scalability, making it applicable to larger datasets and real-time educational settings.

Acknowledgement: None.

Funding Statement: This work is supported by National Natural Science Foundation of China (Nos. 62266054 and 62166050), Key Program of Fundamental Research Project of Yunnan Science and Technology Plan (No. 202201AS070021), Yunnan Fundamental Research Projects (No. 202401AT070122), Yunnan International Joint Research and Development Center of China-Laos-Thailand Educational Digitalization (No. 202203AP140006) and Scientific Research Foundation of Yunnan Provincial Department of Education (No. 2024Y159).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Jun Wang, Mingjie Wang, Zijie Li; data collection: Zijie Li; analysis and interpretation of results: Zijie Li, Ken Chen, Jiatian Mei; draft manuscript preparation: Jun Wang, Mingjie Wang, Zijie Li, Shu Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are openly available in assistment-2009-2010-data at https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data (accessed on 01 October 2023).

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] G. Abdelrahman, Q. Wang, and B. Nunes, "Knowledge tracing: A survey," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–37, Nov. 2023. doi: 10.1145/3569576.

[2] A. Osika, S. Nilsson, A. Sydorchuk, F. Sahin, and A. Huss, "Second language acquisition modeling: An ensemble approach," in *Proc. Thirteenth Workshop Innov. Use NLP Build. Educ. Appl.*, J. Tetreault, J. Burstein, E. Kochmar, C. Leacock, H. Yannakoudakis, Eds., New Orleans, LA, USA: Association for Computational Linguistics, Jun. 2018, pp. 217–222. doi: 10.18653/v1/W18-0525

[3] A. Vaswani *et al.*, "Attention is all you need," presented at NIPS'17: *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec 7–10, 2017.

[4] I. O. Tolstikhin *et al.*, "MLP-Mixer: An all-MLP architecture for vision," *Adv. Neural Inf. Process Syst.*, vol. 34, pp. 24261–24272, 2021.

[5] M. Li *et al.*, "MLP4Rec: A pure MLP architecture for sequential recommendations," presented at National Joint Conf. Artif. Intell. (IJCAI 2022), Messe Wien, Vienna, Austria, Jul. 23–29, 2022.

[6] F. Mai *et al.*, "Hypermixer: An MLP-based low cost alternative to transformers," 2022, *arXiv:2203.03691*.

[7] C. Piech *et al.*, "Deep knowledge tracing," presented at NIPS'15: Proc. 28th Int. Conf. Neural Inf. Process. Syst., Montreal, QC, Canada, Dec. 7–12, 2015.

[8] C. -K. Yeung and D. -Y. Yeung, "Addressing two problems in deep knowledge tracing via prediction-consistent regularization," in *Proc. Fifth Annu. ACM Conf. Learning Scale*, London, UK, 2018, pp. 1–10.

[9] K. Nagatani, Q. Zhang, M. Sato, Y. -Y. Chen, F. Chen and T. Ohkuma, "Augmenting knowledge tracing by considering forgetting behavior," in *The World Wide Web Conf.*, 2019, pp. 3101–3107.

[10] Z. Liu *et al.*, "Enhancing deep knowledge tracing with auxiliary tasks," in *Proc. ACM Web Conf. 2023*, Austin, TX, USA, 2023, pp. 4178–4187.

[11] S. Shen *et al.*, "Learning process-consistent knowledge tracing," in *Proc. 27th ACM SIGKDD Conf. Knowledge Discovery Data Mining*, Singapore, 2021, pp. 1452–1460.

[12] J. Chen, Z. Liu, S. Huang, Q. Liu, and W. Luo, "Improving Interpretability of Deep Sequential Knowledge Tracing Models with Question-centric Cognitive Representations," Mar. 16, 2023, *arXiv:2302.06885*.

[13] S. Pandey and G. Karypis, "A self-attentive model for knowledge tracing," Jul. 01, 2019, *arXiv:1907.06837*.

[14] Y. Choi *et al.*, "Towards an appropriate query, key, and value computation for knowledge tracing," in *Proc. of the Seventh ACM Conf. on Learning @ Scale*, 2020, pp. 341–344.

[15] A. Ghosh, N. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," in *Proc. 26th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, CA, USA, 2020, pp. 2330–2339.

[16] H. Nakagawa, Y. Iwasawa, and Y. Matsuo, "Graph-based knowledge tracing: Modeling student proficiency using graph neural network," in *IEEE/WIC/ACM Int. Conf. on Web Intelligence*, Thessaloniki, Greece, 2019, pp. 156–163.

[17] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen and A. Mian, "Bi-CLKT: Bi-graph contrastive learning based knowledge tracing," *Knowl.-Based Syst.*, vol. 241, 2022, Art. no. 108274. doi: 10.1016/j.knosys.2022.108274.

[18] W. Lee, J. Chun, Y. Lee, K. Park, and S. Park, "Contrastive learning for knowledge tracing," in *Proc. ACM Web Conf. 2022*, 2022, pp. 2330–2338.

[19] M. Li *et al.*, "AutoMLP: Automated MLP for sequential recommendations," in *Proc. ACM Web Conf. 2023*, Apr. 2023, pp. 1190–1198. doi: 10.1145/3543507.3583440.

[20] X. Xiong, S. Zhao., E. G. Van Inwegen, and J. E. Beck, "Going deeper with deep knowledge tracing," in *Proc. 9th Int. Conf. Educational Data Mining*, Raleigh, USA, 2016, pp. 545–550.

[21] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. J. Gordon, K. R. Koedinger and I. Algebra, "Algebra I 2005–2006," 2010. Accessed: Jun. 1, 2012. [Online]. Available: http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp

[22] Z. Liu, Q. Liu, J. Chen, S. Huang, J. Tang and W. Luo, "pyKT: A python library to benchmark deep learning based knowledge tracing models," in *Adv. Neural Inf. Process. Syst. 35 (NeurIPS 2022)*, New Orleans, LA, USA, Nov. 28–Dec. 9, 2022.

[23] J. Lee and D. -Y. Yeung, "Knowledge query network for knowledge tracing: How knowledge interacts with skills," in *Proc. 9th Int. Conf. Learning Analytics Knowledge*, Tempe, AZ, USA, ACM, Mar. 2019, pp. 491–500. doi: 10.1145/3303772.3303786.

[24] X. Guo, Z. Huang, J. Gao, M. Shang, M. Shu and J. Sun, "Enhancing knowledge tracing via adversarial training," Aug. 09, 2021, *arXiv:2108.04430*.