ANT-KT:Adaptive NAS Transformers for Knowledge Tracing

Anonymous ACL submission

Abstract

Knowledge tracing aims to assess students' mastery of knowledge points in real-time. In recent years, researchers have attempted to introduce neural architecture search (NAS) into knowledge tracing tasks to automatically design more efficient network structures. However, existing methods suffer from large search spaces and long search times, limiting their practical application value. To address these issues, this paper proposes an adaptive neural architecture search architecture based on Transformer for knowledge tracing tasks, named ANT-KT, with two improvement schemes. First, an optimized decoder is designed, which improves model performance while reducing time and space complexity through a novel linear attention mechanism, enhancing the model's ability to capture long-range dependencies. Second, a variant encoder is designed, which captures both local and global dependencies of the input sequence by introducing Mamba and combining it with learnable state vectors and convolution operations. Experiments on two large-scale real-world datasets, EdNet and RAIEd2020, show that the proposed method achieves certain improvements on three metrics and reduces the search time by 7%.

1 Introduction

Knowledge tracing (Abdelrahman et al., 2023) aims to assess students' mastery of knowledge points in real-time and is one of the key links in realizing personalized adaptive learning(Dowling and Hockemeyer, 2001). With the rapid development of online education platforms(Geigle and Zhai, 2017)(Anderson et al., 2014), the massive amount of learning record data provides a brand new opportunity to build knowledge tracing models. By modeling students' question-answering sequences, knowledge tracing can dynamically predict students' knowledge states, providing important basis for downstream tasks such as learning path planning and learning material recommendation. Traditional knowledge tracing methods are mainly based on Bayesian networks and factor analysis, but their representation ability is limited. In recent years, deep learning has made significant progress in the field of knowledge tracing. Conversely, existing models still have deficiencies in capturing students' forgetting behaviors, mining structural information between questions or knowledge points, etc. Therefore, exploring more effective knowledge tracing models is of great significance for improving the intelligence level of adaptive learning systems.

Research on knowledge tracing dates back to the 1990s with the introduction of the Bayesian Knowledge Tracing (BKT) model (Corbett and Anderson, 2005). Subsequently, various probabilistic graphical models were proposed, including dynamic Bayesian networks (Yudelson et al., 2013) and hidden Markov models (Baker et al., 2010). While these models described student knowledge state transitions using predefined rules, they had limitations in representation and generalization. Deep learning has since revolutionized knowledge tracing. The Deep Knowledge Tracing (DKT) model (Piech et al., 2015) pioneered the use of RNNs to model student interactions as sequences. Dynamic Key-Value Memory Network (DKVMN) (Zhang et al., 2017) further improved this by introducing memory mechanisms to capture dynamic knowledge state changes. The Self-Attentive Knowledge Tracing (SAKT) model (Pandey and Karypis, 2019) employed self-attention to model inter-question relationships. Meanwhile, the Separated Self-Attentive Neural Knowledge Tracing (SAINT) model (Choi et al., 2020a) marked a significant

leap. It applied Transformers to knowledge tracing, achieving substantial performance improvements through complex exercise interaction modeling. Despite these advancements in characterizing learning processes and knowledge point relationships, most methods relied on fixed network structures, overlooking the potential of architecture optimization. Recent work (Yang et al., 2023) has explored applying Neural Architecture Search (NAS)(Elsken et al., 2019) to Transformer-based knowledge tracing models, aiming for automatic optimal architecture discovery. However, the computational demands and time requirements of NAS methods present challenges for efficient and practical application in knowledge tracing.

To address the limitations of applying NAS to knowledge tracing tasks, this paper proposes a Transformer variant for neural architecture search, with its core containing two innovative modules: an enhanced encoder and an optimized decoder. The enhanced encoder aims to improve the model's ability to extract local features of sequences. By introducing the Mamba module and combining it with learnable state vectors and convolution operations, it achieves precise modeling of local and global dependencies, thereby better capturing short-term changes in students' knowledge states and small progress in the learning process. The optimized decoder focuses on optimizing the fusion of global information. We design a novel attention computation mechanism that enhances the model's ability to capture long-range dependencies while improving performance, helping the model more accurately understand and predict changes in students' knowledge states during long-term learning. The main contributions of this paper are summarized as follows:

1. Proposes ANT-KT, an adaptive neural architecture search architecture based on Transformer for knowledge tracing tasks. ANT-KT incorporates an enhanced encoder and an optimized decoder to improve the model's ability to capture changes in students' knowledge states and details of the learning process;

2. Designs an enhanced encoder to improve the model's ability to extract local features of sequences. This encoder introduces the Mamba module and combines it with learnable state vectors and convolution operations to achieve precise modeling of local and global dependencies. This design enables the model to better capture short-term changes in students' knowledge states and small progress in the learning process;

3. Proposes an optimized decoder that focuses on optimizing the fusion of global information. A novel attention computation mechanism is designed to adaptively fuse the encoder's output into each position of the decoder. This mechanism enhances the model's ability to capture long-range dependencies, enabling it to more accurately understand and predict changes in students' knowledge states during long-term learning;

4. Conducts extensive experiments on two large-scale real-world datasets, EdNet and RAIEd2020, and compares with multiple existing methods. The experimental results show that the proposed method achieves certain improvements on multiple evaluation metrics and reduces the search time by 7%, demonstrating significant performance improvement and efficiency advantages.

2 Related Work

Transformers in Knowledge Tracing: The application of Transformers and their improved models in knowledge tracing has been introduced into the field of knowledge tracing in recent years as a powerful sequential modeling tool (Vaswani et al., 2017). SAINT+ (Shin et al., 2021) builds an encoder-decoder structure based on Transformers and realizes the mining of question interaction information through self-attention mechanisms. HAKT (Sha et al., 2021) further introduces hierarchical attention on this basis to characterize the dependencies between knowledge points. However, existing work is mainly limited to the original Transformer architecture, ignoring the latest progress in structural optimization.

NAS in Knowledge Tracing: Despite the performance improvements achieved by deep knowledge tracing models in predicting students' learning states, the architectures of these

models are mostly designed manually, which cannot guarantee to obtain the globally optimal structure. To further enhance model performance and reduce the complexity of manual design, researchers have begun to explore the application of Neural Architecture Search (NAS)(Elsken et al., 2019) techniques in the field of knowledge tracing. Ding et al. pioneered the application of NAS to the knowledge tracing task, proposing the NAS-Cell method (Ding and Larson, 2020). This method utilizes reinforcement learning to search for the cell structure of LSTMs, ultimately obtaining a model architecture that outperforms manually designed ones and achieving significant performance improvements in multiple knowledge tracing tasks. This study demonstrates the initial application and potential value of NAS in the knowledge tracing domain. Yang et al. further extended the application of NAS in the knowledge tracing field, proposing ENAS-KT (Yang et al., 2023), a more comprehensive and in-depth neural architecture search method, and designing a search space structure specialized for knowledge tracing.

3 Methodology

3.1 Knowledge Tracing Problem Definition

Considering a student's response history to a set of exercises, the knowledge tracing (KT) task aims to predict the probability of the student answering new exercises correctly. Formally, a student's activities are recorded as a sequence of n interactions $I = \{I_1, \ldots, I_n\}$, where the i-th interaction is represented as a triplet $I_i = (X_i, T_i, R_i)$. X_i contains information about the exercise itself and related metadata (such as question type, involved knowledge points, etc.), T_i represents the time taken by the student to answer the question, and $R_i \in \{0, 1\}$ represents the student's answer ($R_i = 1$ indicates a correct answer, $R_i = 0$ indicates an incorrect answer).

Given the data of a student's first n interactions $I = \{I_1, \ldots, I_n\}$, the goal of the knowledge tracing task is to predict the probability of the student answering the $(n+1)$-th exercise X_{n+1} correctly, as follows:

$$P(R_{n+1}|I_1, I_2, \ldots, I_n, X_{n+1}). \quad (1)$$

3.2 Overall Framework

This paper proposes a Transformer variant structure suitable for knowledge tracing tasks. As shown in Figure 1, the overall framework consists of three parts: 1) a selective hierarchical input module for extracting and selecting feature inputs of different granularities and types; 2) an enhanced encoder that captures complex dependencies in the exercise sequence based on the Mamba module and hybrid operations; 3) an optimized decoder that introduces a linear attention and selective output to accurately predict knowledge states using encoder information.

3.2.1 Selective Hierarchical Input Module

For a student's interaction sequence $I = \{I_1, \ldots, I_n\}$, where each interaction $I_i = (X_i, T_i, R_i)$, these features are first mapped to a set of candidate embeddings $X_{embed} = \{embed_i \in \mathbb{R}^{L \times d} \mid 1 \leq i \leq Num\}$, where d is the embedding dimension and Num is the number of features. The appropriate embeddings are then selected as input to the encoder and decoder through two binary vectors b_{En} and b_{De}:

$$X_{En} = \{e_i \in X_{embed} \mid b_i = 1, b_i \in b_{En}\}, \quad (2)$$
$$X_{De} = \{e_i \in X_{embed} \mid b_i = 1, b_i \in b_{De}\}, \quad (3)$$

where $b_{En}, b_{De} \in \{0, 1\}^{1 \times Num}$, ensuring that at least one embedding is selected. This flexible selection mechanism allows the model to automatically discover the utility of different feature combinations according to task requirements.

3.2.2 Enhanced Encoder

The enhanced encoder proposed in this paper introduces various improvement mechanisms on the basis of Transformers to enhance the model's feature extraction and representation learning capabilities in knowledge tracing tasks. The encoder stacks N_E basic units to model the student's interaction sequence.

Specifically, let H^i_{En} denote the output of the i-th encoder layer, and $H^0_{En} = X_{En}$ be the initial input of the encoder. Consider the i-th encoder unit, whose computation process is:

$$Z^i_{En} = \text{Mamba}(H^{i-1}_{En}), \quad (4)$$
$$H^{i-1}_{En} = Z^i_{En} + M^i_{En}(\text{LN}(\text{FFN}(Z^i_{En})), \quad (5)$$

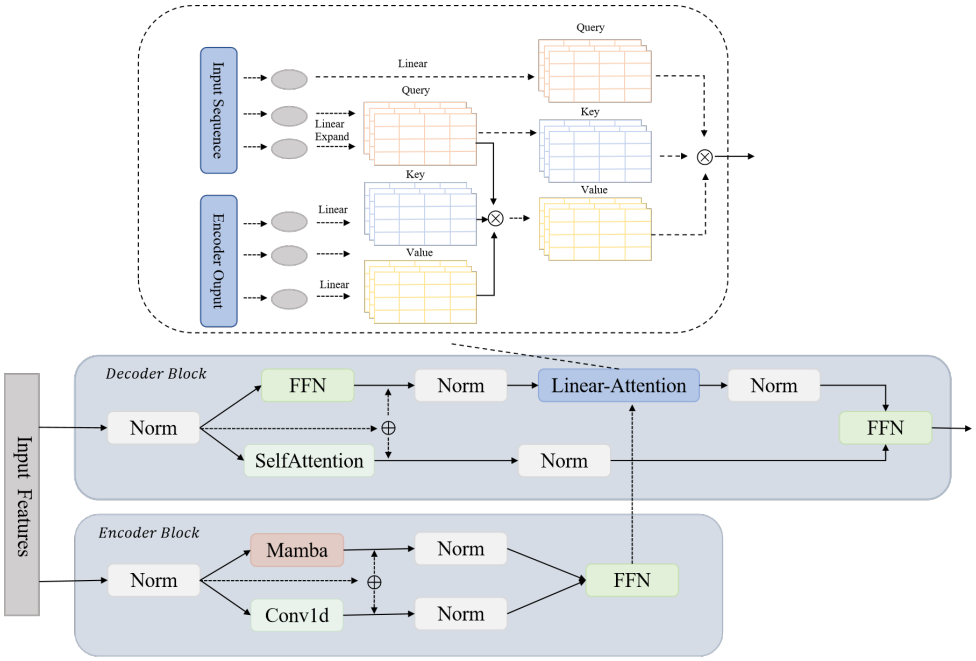

Figure 1: The architecture of the Transformer variant proposed in this paper is described. The encoder processes input features to capture local and global dependencies, while the decoder receives the output of the encoder and generates predicted probabilities of knowledge states.

where M^i_{En} represents the hybrid operation of the i-th encoder layer, LN represents layer normalization, and FFN represents the feed-forward network.

Specifically, the hybrid operation M^i_{En} is defined as a weighted combination of multiple candidate operations:

$$M^i_{En}(x) = \sum_{k=1}^{K} \alpha^i_k o_k(x), \tag{6}$$

where o_k represents the k-th candidate operation (such as self-attention, gated convolution, etc.) and α^i_k is the corresponding mixing coefficient. The purpose of introducing hybrid operations is to enhance the model's feature extraction ability and generalization by adaptively combining different types of feature interactions, enabling the model to flexibly adapt to different types of sequence patterns.

First, we extract input features H^{i-1}_{En} through one-dimensional convolution, and then add them to the original input to obtain local features C^i_{en}:

$$C^i_{en} = H^{i-1}_{En} + \text{Conv1d}(H^{i-1}_{En}). \tag{7}$$

To avoid gradient vanishing and ensure training stability, the encoder unit introduces residual connection normalization to ensure the consistency of feature processing:

$$H^i_{En} = \text{LayerNorm}(H^{i-1}_{En} + M^i_{En}(C^i_{en})). \tag{8}$$

Finally, the final feature representation is obtained after processing by the feed-forward network:

$$H^i_{En} = H^i_{En} + M^i_{En}(\text{FFN}(H^i_{En})). \tag{9}$$

3.2.3 Optimized Decoder

The goal of the decoder is to predict the student's answer to the question using the output information $H^{N_E}_{En}$ of the encoder. The decoder stacks N_D basic units, each unit containing three sub-layers: self-attention layer, linear attention layer, and feed-forward network layer.

Let H^j_{De} denote the output of the j-th decoder layer, and $H^0_{De} = X_{De}$ be the initial input of the decoder. Consider the j-th decoder unit, whose computation process is:

Layer normalization is performed on the input H^{j-1}_{De} to ensure numerical stability:

$$H^j_{De} = \text{LayerNorm}(H^{j-1}_{De}). \tag{10}$$

Then, self-attention is used to capture the interdependencies between elements in the sequence:

$$Z_{De}^{j} = H_{De}^{j} + \text{SelfAttn}(H_{De}^{j}). \quad (11)$$

Next, to reduce computational complexity and to handle long sequence problems, we design a linear attention for information interaction between the decoder and encoder, modeling long-range dependencies with linear complexity. Specifically:

First, the output Z_{De}^{j} of the decoder and the output $H_{En}^{N_E}$ of the encoder are combined, and query, key, and value matrices are generated through linear transformation:

$$A = W_a[H_{En}^{N_E}; Z_{De}^{j}] \in \mathbb{R}^{L\times 2d}, \quad (12)$$

$$B = W_b H_{En}^{N_E} \in \mathbb{R}^{L\times d}, \quad (13)$$

where $W_a \in \mathbb{R}^{2d\times 2d}$ and $W_b \in \mathbb{R}^{d\times d}$ are linear transformation matrices, and ; represents the concatenation operation. Through these matrices, the model extracts global information from the encoder's output.

The query and key matrices are dot-produced and scaled, then a nonlinear activation operation is performed to obtain the attention weights:

$$E = \text{ReLU}(\frac{W_e A^T}{\sqrt{d}}) \in \mathbb{R}^{d\times L}, \quad (14)$$

where $W_e \in \mathbb{R}^{d\times 2d}$ is a linear transformation matrix. By scaling the dot product and applying an activation function, the model can assign different weights to different feature dimensions. The value matrix B is weighted summed using the attention weights E to generate the feature representation of linear attention:

$$F = \text{ReLU}(EB) \in \mathbb{R}^{d\times d}. \quad (15)$$

Through weighted summation, the encoder's global information is effectively propagated to each position of the decoder. Finally, the result F of the weighted summation is combined with the decoder's current output Z_{De}^{j}, and the final feature representation is generated through linear transformation:

$$\widetilde{Z}De^{j} = ZDe^{j} + \text{ReLU}(W_f F Z_{De}^{j}), \quad (16)$$

where $W_f \in \mathbb{R}^{d\times d}$ is a linear transformation matrix. This process preserves local information in the decoder and integrates the encoder's global information via linear attention, improving long-range dependency modeling. After linear attention, local convolution extracts features, which are further processed with hybrid operations:

$$H_{De}^{j} = \widetilde{Z}De^{j} + MDe^{j}(\text{Conv1d}(\widetilde{Z}_{De}^{j})). \quad (17)$$

The hybrid operation allows the decoder to select the optimal convolution kernel size, enhancing the flexibility of feature extraction.

The last layer of the decoder further processes the decoder's output through the feed-forward network of the hybrid operation, whose computation process is as follows:

$$H_{De}^{j} = H_{De}^{j} + M_{De}^{j}(\text{FFN}(H_{De}^{j})). \quad (18)$$

Finally, the answer prediction generates the probability of the student's answer through linear transformation and the Sigmoid function:

$$R = \sigma(w_r H_{De}^{N_D} + b_r), \quad (19)$$

w_r and b_r are the weights and biases of the fully connected layer.

3.3 Neural Architecture Search

To find the optimal network configuration, define the above model as a flexible search space, as shown in Figure 2. And apply NAS for optimization. The search space contains the following three groups of key parameters:

Input feature selection vectors: $b_{En}, b_{De} \in 0, 1^{1\times Num}$, where Num is the total number of candidate features, controlling which input features will be used for the encoder and decoder.

Hybrid operation parameters: The hybrid operation parameters of the encoder and decoder are $\alpha_{En} \in \mathbb{R}^{N_E\times K}$ and $\alpha_{De} \in \mathbb{R}^{N_D\times K}$, where K is the number of candidate operations. Different operations (such as convolution, feed-forward network, etc.) in each hybrid operation are weighted and combined through weights α, defining the final structure of the model.

Number of layers selection: The number of encoder layers N_E and the number of decoder layers N_D.

3.3.1 Objective Function

Let θ be all the weight parameters of the model and $\mathcal{A}$ represent the search space, then the goal

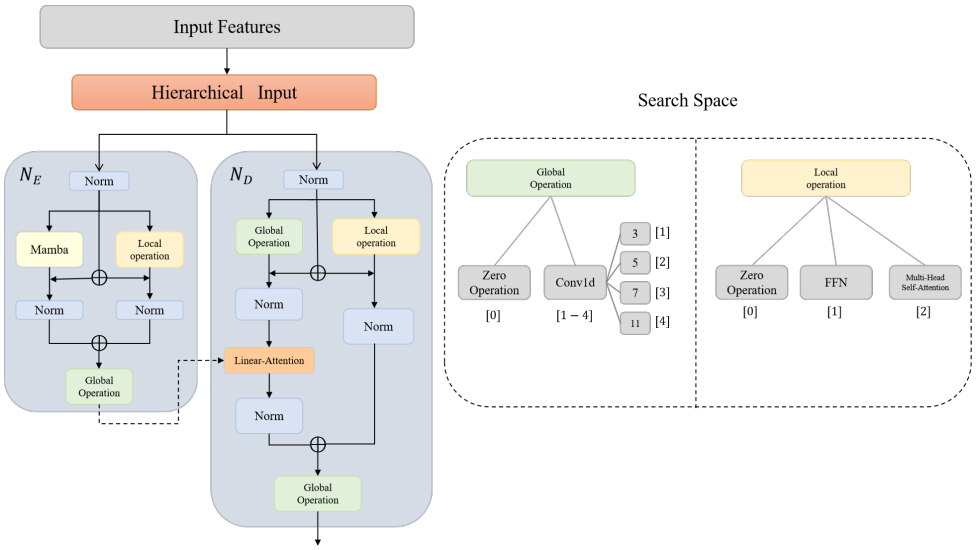

Figure 2: The proposed search space.

of joint learning can be expressed as:

$$\min_{a \in A} \mathcal{L}_{val}(w^*(\alpha), \alpha), \tag{20}$$

where $\mathcal{L}train$ and $\mathcal{L}val$ represent the loss functions of the model on the training set and validation set, respectively, α represents the architecture parameters in the search space, and $w^*(\alpha)$ represents the optimal weights obtained by training with the fixed architecture α.

3.3.2 Search Strategies

The search process employs two key strategies:

1. **Weighted Combination of Operations**: Each operation o^i is assigned an importance weight α_i:

$$o(x) = \sum_{i=1}^{K} \frac{\exp(\alpha_i)}{\sum_{j=1}^{K} \exp(\alpha_i)} o^i(x). \tag{21}$$

   This allows flexible combination of different structural elements, improving generalization.

2. **Weight Sharing**: Certain parameters are shared across different architectural configurations, significantly reducing computational overhead.

These strategies work in tandem to explore a wide range of model structures while maintaining computational feasibility.

3.3.3 Alternating Optimization

To efficiently optimize the architecture parameters α and weight parameters w, we adopt the gradient-based optimization method DARTS. The core idea of DARTS is to relax the discrete architecture selection into a continuous parameter mixture, so that gradient information can be used to optimize the architecture parameters and weight parameters simultaneously. In each training step, data D_{train} is first sampled from the training set, and the gradient is calculated based on the current architecture parameters α_t to update the weight parameters:

$$w_{t+1} = w_t - \eta_w \nabla_w \mathcal{L}train(w_t, \alpha_t, Dtrain). \tag{22}$$

Then, data D_{val} is sampled from the validation set, and the gradient is calculated based on the updated weight parameters w_{t+1} to update the architecture parameters:

$$\alpha_{t+1} = \alpha_t - \eta_\alpha \nabla_\alpha \mathcal{L}val(wt+1, \alpha_t, D_{val}), \tag{23}$$

where η_w and η_α represent the learning rates of weights and architecture, respectively. Through alternating optimization, the model can continuously update the architecture to adapt to the current task during the training process. After the search is over, for each position that requires a decision, we select the operation with the largest α value as the final architecture:

$$o^* = \arg\max_{\alpha_i} o^{(i)}. \tag{24}$$

The operation with the largest weight α_i is selected as the optimal architecture α^*. After

Datasets/Statistics	# of Interactions (# of Tags)	# of Students (# of Tag-Sets)	# of Exercises (# of Bundles)	# of Skills (# of Explanations)
EdNet	95,293,926 (302)	84,309 (1,792)	13,169 (9,534)	7 (-)
RAIEd2020	99,271,300 (189)	393,656 (1,520)	13,523 (-)	7 (2)

Table 1: Statistics of the two largest education datasets for knowledge tracing, EdNet and RAIEd2020.

fixing the optimal architecture α^*, the model weights are retrained on the complete dataset to obtain the final optimized model. Through the above steps, the design space of the Transformer architecture can be explored efficiently and comprehensively, and the optimal architecture suitable for the knowledge tracing task can be automatically discovered.

4 Experiments

To comprehensively evaluate the effectiveness of our proposed model on the knowledge tracing task, this section will introduce the experimental setup, result analysis, and further ablation studies in detail. We validated the model performance on two large-scale knowledge tracing datasets and compared it with multiple strong baselines.

4.1 Datasets

To rigorously evaluate the proposed approach, experiments utilized the two most extensive and challenging real-world educational datasets currently available: EdNet(Choi et al., 2020b) and RAIEd2020(Howard et al., 2020). Table 1 summarizes the key characteristics of these datasets.

The datasets include feature embeddings for exercises, skills, tags, tag-sets, and bundles/explanations, as shown in Table 1. In addition to these five embeddings, the candidate input embedding X_{embed} incorporates seven additional feature embeddings, for a total of 12 embeddings: Answer embedding , Continuous elapsed time embedding, Categorical elapsed time embedding in seconds , Continuous lag time embedding, Categorical lag time embeddings in seconds, minutes, and days .

By leveraging these large-scale, real-world datasets with rich feature embeddings, the experiments provide a convincing validation of the proposed methodology in authentic educational contexts.

4.2 Baseline Models

We compared our model with the following knowledge tracing models: DKT (Piech et al., 2015), HawkesKT (Wang et al., 2021), CT-NCM (Ma et al., 2022), SAKT (Pandey and Karypis, 2019), AKT (Ghosh et al., 2020), SAINT (Choi et al., 2020a), SAINT+ (Shin et al., 2021), NAS-Cell (Ding and Larson, 2020), and ENAS-KT (Yang et al., 2023).

Model	RMSE ↓	ACC ↑	AUC ↑
DKT	0.4653	0.6537	0.6952
HawkesKT	0.4475	0.6888	0.7487
CT-NCM	0.4364	0.7063	0.7743
SAKT	0.4405	0.6998	0.7680
AKT	0.4399	0.7016	0.7686
SAINT	0.4322	0.7132	0.7825
SAINT+	0.4285	0.7188	0.7916
NAS-Cell	0.4345	0.7143	0.7796
ENAS-KT	0.4209	0.7295	0.8062
Ours	**0.4062**	**0.7527**	**0.8355**
Improve[1]	**(-3.18%)**	**(3.18%)**	**(3.63%)**

Table 2: Comparison of various models on the EdNet dataset.

Model	RMSE ↓	ACC ↑	AUC ↑
DKT	0.4632	0.6622	0.7108
HawkesKT	0.4453	0.6928	0.7525
CT-NCM	0.4355	0.7079	0.7771
SAKT	0.4381	0.7035	0.7693
AKT	0.4368	0.7076	0.7752
SAINT	0.4310	0.7143	0.7862
SAINT+	0.4272	0.7192	0.7934
NAS-Cell	0.4309	0.7167	0.7839
ENAS-KT	0.4196	0.7313	0.8089
Ours	**0.4122**	**0.7438**	**0.8239**
Improve	**(-1.76%)**	**(1.71%)**	**(1.85%)**

Table 3: Comparison of various models on the RAIEd2020 dataset.

4.3 Implementation Details

All students are randomly divided into 70%/10%/20% for training/validation/testing.

[1]Calculation: (Ours - previous best) / previous best

The maximum length of the input sequence is set to 100, and we truncate student learning interactions exceeding 100 into several subsequences and use 5-fold cross-validation. The number of blocks N=4, the embedding size D=128, and the hidden dimension of FFN is set to 128. To train the supernet, the number of epochs, learning rate, dropout rate, and batch size are set to 60, 1e-3, 0.1, and 128, respectively. All experiments were performed on a single RTX3090.

4.4 Experimental Results

We use AUC, accuracy (ACC), and root mean square error (RMSE) as evaluation metrics. Table 23 shows the performance comparison of different models on the two datasets. It can be seen that ANT-KT achieves the best results on all datasets and evaluation metrics. Compared with the optimal baseline, ACC, AUC, and RMSE are improved by an average of 2.32%, 2.93%, and 1.47%, respectively, demonstrating the superiority of ANT-KT in modeling students' learning processes and predicting future performance. Considering the volatility of the graphics card and the difference in methods, for the comparison of training time, we selected ENAS-KT, which also uses the NAS method for the knowledge tracing task, and selected the one-fold of cross-validation where each epoch is smooth as a comparison. The training time improved from 41.86 hours to 39.29 hours, saving 7%.

4.5 Ablation Studies

To assess the effectiveness of key components in our model, we conducted ablation experiments focusing on the Mamba module in the enhanced encoder and the linear attention in the optimized decoder. The results are presented in Table 4.

The linear attention demonstrated significant improvements across all metrics on both datasets. This validates its superiority in capturing global dependencies within sequences, enabling more accurate modeling of students' knowledge state evolution.

Conversely, the Mamba module alone showed limited impact, leading to performance declines. This suggests that in knowledge tracing tasks, global semantic modeling may be more crucial than local feature extraction.

Dataset	Metric	Baseline	+Mamba	+Attn	ANT-KT
EdNet	RMSE	0.4209	0.4246	0.4200	**0.4062**
	ACC	0.7295	0.7243	0.7424	**0.7527**
	AUC	0.8062	0.7993	0.8208	**0.8355**
RAIEd2020	RMSE	0.4196	0.4275	0.4147	**0.4122**
	ACC	0.7313	0.7198	0.7408	**0.7438**
	AUC	0.8089	0.7939	0.8201	**0.8239**

Table 4: Performance of ablation studies

Interestingly, combining the Mamba module with linear attention produced a synergistic effect, outperforming the model with linear attention alone across all metrics. This unexpected performance boost can be attributed to the complementary strengths of Mamba's local feature modeling and linear attention's global information aggregation. In summary, this study highlights the importance of synergy between linear attention and the Mamba module. While linear attention is crucial for capturing global dependencies, its combination with Mamba leads to substantial performance gains. This finding suggests that future knowledge tracing models should explore the potential of collaboratively modeling both global and local information to achieve comprehensive understanding of students' learning processes.

5 Conclusion

This paper proposes a NAS method ANT-KT based on Transformers for knowledge tracing tasks, aiming to solve the deficiencies of existing methods in search space and time efficiency. By designing an enhanced encoder and an optimized decoder, ANT-KT has significantly improved the ability to capture short-term and long-term dependencies and has made significant improvements in capturing students' short-term changes in knowledge state and details of the learning process. The enhanced encoder combines the Mamba module, learnable state vectors, and convolution operations to improve the modeling ability of local and global dependencies in sequences, while the optimized decoder optimizes the fusion of global information through a novel linear attention.

References

Ghodai Abdelrahman, Qing Wang, and Bernardo Nunes. 2023. Knowledge tracing: A survey.

ACM Comput. Surv., 55(11).

Ashton Anderson, Daniel Huttenlocher, Jon Kleinberg, and Jure Leskovec. 2014. Engaging with massive online courses. In *Proceedings of the 23rd International Conference on World Wide Web*, page 687–698.

Ryan S. J. d. Baker, Albert T. Corbett, Sujith M. Gowda, Angela Z. Wagner, Benjamin A. MacLaren, Linda R. Kauffman, Aaron P. Mitchell, and Stephen Giguere. 2010. Contextual slip and prediction of student performance after use of an intelligent tutor. In *User Modeling, Adaptation, and Personalization*, pages 52–63.

Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. 2020a. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the Seventh ACM Conference on Learning @ Scale*, page 341–344.

Youngduck Choi, Youngnam Lee, Dongmin Shin, Junghyun Cho, Seoyon Park, Seewoo Lee, Jineon Baek, Chan Bae, Byungsoo Kim, and Jaewe Heo. 2020b. Ednet: A large-scale hierarchical dataset in education. In *Artificial Intelligence in Education*, pages 69–73.

Albert T. Corbett and John R. Anderson. 2005. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4:253–278.

Xinyi Ding and Eric C. Larson. 2020. Automatic rnn cell design for knowledge tracing using reinforcement learning. In *Proceedings of the Seventh ACM Conference on Learning @ Scale*, page 285–288.

C.E. Dowling and C. Hockemeyer. 2001. Automata for the assessment of knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 13(3):451–461.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: a survey. *J. Mach. Learn. Res.*, 20:1997–2017.

Chase Geigle and ChengXiang Zhai. 2017. Modeling mooc student behavior with two-layer hidden markov models. In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, page 205–208.

Aritra Ghosh, Neil Heffernan, and Andrew S. Lan. 2020. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2330–2339.

Addison Howard, bskim90, Cheehyun Lee, Dongmin (Min) Shin, Hoon Pyo (Tim) Jeon, Jineon (Jin) Baek, Karis Chang, kiyoonkay, N. Heffernan, seonwooko, Sohier Dane, and Yohan Lee. 2020. Riiid answer correctness prediction.

Haiping Ma, Jingyuan Wang, Hengshu Zhu, Xin Xia, Haifeng Zhang, Xingyi Zhang, and Lei Zhang. 2022. Reconciling cognitive modeling with knowledge forgetting: A continuous time-aware neural network approach. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2174–2181.

Shalini Pandey and George Karypis. 2019. A self-attentive model for knowledge tracing. *arXiv preprint arXiv:1907.06837*.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems*, volume 28.

Xiao Sha, Zhu Sun, and Jie Zhang. 2021. Hierarchical attentive knowledge graph embedding for personalized recommendation. *Electronic Commerce Research and Applications*, 48:101071.

Dongmin Shin, Yugeun Shim, Hangyeol Yu, Seewoo Lee, Byungsoo Kim, and Youngduck Choi. 2021. Saint+: Integrating temporal features for ednet correctness prediction. In *LAK21: 11th International Learning Analytics and Knowledge Conference*, page 490–496.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Chenyang Wang, Weizhi Ma, Min Zhang, Chuancheng Lv, Fengyuan Wan, Huijie Lin, Taoran Tang, Yiqun Liu, and Shaoping Ma. 2021. Temporal cross-effects in knowledge tracing. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, page 517–525.

Shangshang Yang, Xiaoshan Yu, Ye Tian, Xueming Yan, Haiping Ma, and Xingyi Zhang. 2023. Evolutionary neural architecture search for transformer in knowledge tracing. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*.

Michael V. Yudelson, Kenneth R. Koedinger, and Geoffrey J. Gordon. 2013. Individualized bayesian knowledge tracing models. In *Artificial Intelligence in Education*, pages 171–180.

Jiani Zhang, Xingjian Shi, Irwin King, and Dit-Yan Yeung. 2017. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*, page 765–774.

A Appendix

A.1 Limitations

Firstly, the current study primarily validates the model's performance using two large-scale real-world datasets, predominantly comprising specific types of learning activities and interactions. Therefore, the applicability and performance of the model on different types of educational activities or on smaller, more diverse datasets remain to be further tested. Future work could involve testing the model across a broader range of educational scenarios and diverse datasets to evaluate its generalization capabilities.

Secondly, this research employs NAS in the automated design process of the model, which, while capable of discovering optimized structural configurations, still has room for improvement in terms of the efficiency of the search process and the optimality of the results. Further studies could explore more effective search strategies or refine the existing search algorithms to enhance the efficiency and accuracy of the search process.