ELAKT: Enhancing Locality for Attentive Knowledge Tracing

YANJUN PU, School of Computer Science and Engineering, Beihang University, Beijing, China and Zhongguancun Laboratory, Beijing, China
FANG LIU, Institute of Artificial Intelligence, Beihang University, Beijing, China
RONGYE SHI, Institute of Artificial Intelligence, Beihang University, Beijing, China
HAITAO YUAN, Department of Computer Science and Engineering, Tsinghua University, Beijing, China
RUIBO CHEN, Institute of Artificial Intelligence, Beihang University, Beijing, China
TIANHAO PENG, School of Computer Science and Engineering, Beihang University, Beijing, China
WENJUN WU, Institute of Artificial Intelligence, Beihang University, Beijing, China

Knowledge tracing models based on deep learning can achieve impressive predictive performance by leveraging attention mechanisms. However, there still exist two challenges in attentive knowledge tracing (AKT): First, the mechanism of classical models of AKT demonstrates relatively low attention when processing exercise sequences with shifting knowledge concepts (KC), making it difficult to capture the comprehensive state of knowledge across sequences. Second, classical models do not consider stochastic behaviors, which negatively affects models of AKT in terms of capturing anomalous knowledge states. This article proposes a model of AKT, called Enhancing Locality for Attentive Knowledge Tracing (ELAKT), that is a variant of the deep KT model. The proposed model leverages the encoder module of the transformer to aggregate knowledge embedding generated by both exercises and responses over all timesteps. In addition, it uses causal convolutions to aggregate and smooth the states of local knowledge. The ELAKT model uses the states of comprehensive KCs to introduce a prediction correction module to forecast the future responses of students to deal with noise caused by stochastic behaviors. The results of experiments demonstrated that the ELAKT model consistently outperforms state-of-the-art baseline KT models.

CCS Concepts: • **Applied computing → Education**; • **Information systems → Data mining**; • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Knowledge tracing, self-attention, causal convolution, knowledge aggregation

This work is supported by the State Key Laboratory of Software Development Environment (Funding No. SKLSDE-2020ZX-01) and Zhongguancun Laboratory.
Authors' addresses: Y. Pu, School of Computer Science and Engineering, Beihang University, Xueyuan Road No.37, Beijing 100191, China, and Zhongguancun Laboratory, Beijing, China; e-mail: buaapyj@buaa.edu.cn; F. Liu, R. Shi (Corresponding author), R. Chen, and W. Wu (Corresponding author), Institute of Artificial Intelligence, Beihang University, Beijing, China; e-mails: liufangg@buaa.edu.cn, shirongye@buaa.edu.cn, chenruibo@buaa.edu.cn, wwj09315@buaa.edu.cn; H. Yuan (Corresponding author), Department of Computer Science and Engineering, Tsinghua University, Beijing, China; e-mail: yhaitao45@163.com; T. Peng, School of Computer Science and Engineering, Beihang University, Beijing, China; e-mail: pengtianhao@buaa.edu.cn.

ACM 1046-8188/2024/04-ART112
https://doi.org/10.1145/3652601

ACM Reference Format:
Yanjun Pu, Fang Liu, Rongye Shi, Haitao Yuan, Ruibo Chen, Tianhao Peng, and WenJun Wu. 2024. ELAKT: Enhancing Locality for Attentive Knowledge Tracing. *ACM Trans. Inf. Syst.* 42, 4, Article 112 (April 2024), 27 pages. https://doi.org/10.1145/3652601

1 INTRODUCTION

The **Intelligent Tutoring System** (**ITS**) has become increasingly important in online education over the past decade because it can offer a personalized and adaptive learning experience for a large number of students. A core component of the ITS is the **Knowledge Tracing** (**KT**) model, which can trace the state of latent knowledge of individual students. This information is used to support other components of the system in providing personalized guidance to enable students to quickly achieve their learning outcomes [23, 36, 40].

The KT task can be regarded as a problem of supervised sequential learning. Specifically, given a sequence of the history of a student's historical interactions with exercises, the KT model aims to predict the probability that the student correctly solves the exercise in the next interaction and infer the student's knowledge state during it. Different kinds of KT models have been developed in recent years, and can be roughly divided into two categories: structured KT models and deep KT models [21].

Structured KT methods, such as **Bayesian Knowledge Tracing** (**BKT**) [9], define variables based on the principles of cognitive and educational sciences. This makes it easy for the instructor to interpret the results of the predictions when assessing the student's performance. Early implementations of structured KT methods have limitations in modeling multi-dimensional states of knowledge because the relevant models typically assume that each exercise (or question) a_t encountered at time t is designed for a single knowledge-related concept. Enhanced solutions have been proposed based on the Q-matrix [38], a sparse matrix to measure the cognitive mastery of a concept that associates an exercise a_t to multiple **knowledge concepts** (**KC**). That is, answering a_t correctly requires mastering multiple KCs.

Deep neural network (**DNN**)-based models, such as **deep knowledge tracing** (**DKT**) [34], have recently been developed to solve KT tasks. Early efforts used **recurrent neural networks** (**RNNs**) [29] with **long short-term memory** (**LSTM**) [20] units to model learning by students, and can provide important advantages in predictions without requiring features engineered by a human operator. Attentive KT models have better predictive performance than standard KT models. The attention mechanism in them can capture the relationships between exercises and their relevance to a student's knowledge states [3]. This property enables the model to access any part of the history of the student's states of knowledge, where this makes it suitable for capturing recurring patterns with long-term dependencies.

However, two challenges to **attentive knowledge tracing** (**AKT**) persist. First, in case the historical exercises are diverse, the input embeddings for queries and the keys used in the self-attention mechanism are significantly distinct. The prevalent dot product-based self-attention thus fails to adequately capture the comprehensive knowledge state that is to be mapped to a complete set of KCs, causing the model prone to anomalies [19, 25, 28]. This means that when knowledge embedding is associated only with the current exercise, such as in the KT scenario shown in Figure 1, the comprehensive knowledge state cannot be appropriately captured. Second, the deep KT model does not consider anomalous interactions between students and exercises, that is, the probability of a student guessing or slipping on an exercise [37]. Anomalous interactions within the learning process are referred to as stochastic behaviors in KT. The mismatch between a student's

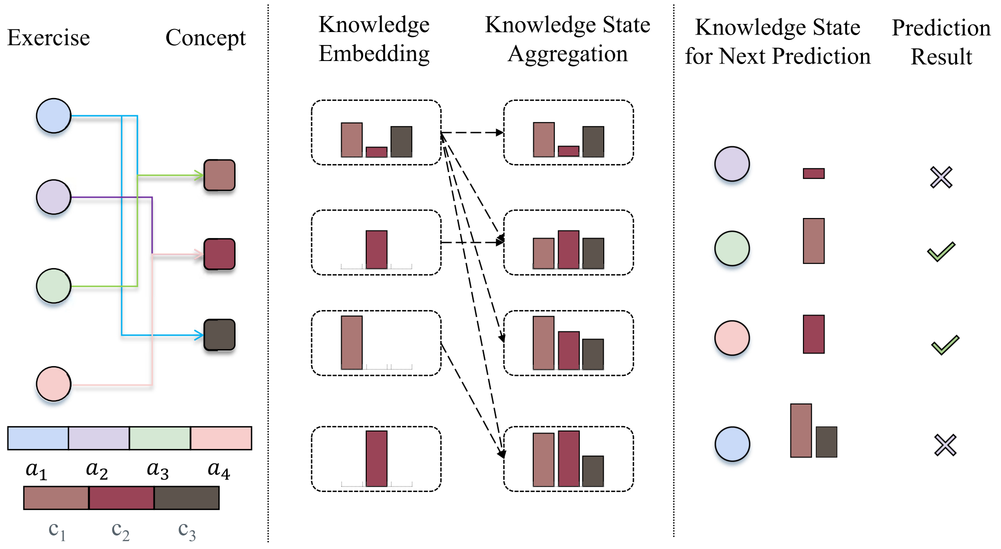


Fig. 1. Illustration of an exercise–concept–knowledge state relationship. It is difficult to model the comprehensive knowledge state at a timestep by using prevalent methods. Knowledge aggregation can help solve the problem of incomplete knowledge states.

response and the corresponding knowledge state leads to an incorrect inference by the self-attention mechanism.

Inspired by memory-enhanced neural networks, some researchers have sought to explicitly store the knowledge states of different concepts by introducing memory modules [2, 6]. This provides ideas for capturing a more accurate knowledge state in the deep KT model. In this article, we aggregate the input knowledge embedding, and further aggregate and smooth hidden knowledge states from the output of the self-attention mechanism to solve the problem of the inability to comprehensively capture the state of knowledge. The comprehensive state of knowledge is mapped to the probability distribution that responds to the complete set of KCs. This probability distribution is then used to predict students' responses and solve the problem of noise caused by stochastic behaviors. Our contributions here are three-fold:

(1) In order to capture more comprehensive knowledge states, we introduce two optional modules for aggregating knowledge embeddings: an encoder module and a causal convolution module.
(2) We further leverage causal convolutions in a second encoder module to aggregate and smooth the local states of knowledge from the output of the self-attention layer such that comprehensive KCs can be obtained and used to address the first challenge above.
(3) Based on the comprehensive KCs obtained above, we introduce a prediction correction module to forecast future exercises' responses and handle the noise caused by stochastic behaviors. The proposed prediction correction module can help tackle the second challenge.

The remainder of this article is organized as follows: Section 2 discusses related work on structured KT and DNN-based KT, and Section 3 describes the preliminary structure of the encoder module and the causal convolution module. Section 4 details the structure of the proposed ELAKT model. In Section 5, we present details of the implementation and the experimental results of the proposed model, and compare it with state-of-the-art deep KT models. Section 6 verifies the effects of aggregating the knowledge embeddings and provides visualizations of the complete KC used for prediction. The final section summarizes the conclusions of this work and discusses future directions of research in the area.

2 RELATED WORK

2.1 Structured Knowledge Tracing Model

A structured KT method, such as the BKT, typically tracks students' knowledge states over time by using the **hidden Markov model** (**HMM**). However, it can track only the students' mastery of a single cognitive skill without specifying the difficulty of the exercises used to assess it. Recent research on the BKT has focused on its extension to assess multiple skills. Brenes and Mostow [13] proposed dynamic cognitive tracing to construct a cognitive model and a student model based on longitudinal data on students. They subsequently introduced **feature-aware student knowledge tracing** (**FAST**) [12] to use different skill-related features to quantify the difficulty of the problem and the student's ability as parameters of the KT model. These feature-based extensions of the BKT heavily rely on expert knowledge when predefining features of the skill and subskills, and do not use techniques of automatic discovery (e.g., the Q-matrix). The **automatic temporal cognitive** (**ATC**) model is an evolution of the **cognitive diagnosis model** (**CDM**) and the KT model. It incorporates multi-dimensional knowledge states and temporal changes, including skill enhancement and forgetting factors. It uses a non-linear state-space framework to encode multi-dimensional levels of the KC and a Q-matrix of the learning items. The ATC model is also capable of deriving values of the Q-matrix from trajectories of student learning in a data-driven approach. Therefore, it is an ideal candidate for governing DNNs in the context of KT in terms of improved interpretability.

2.2 DNN-based Knowledge Tracing Models

2.2.1 DKT and its Extensions. DKT uses RNNs and the LSTM to model students' learning, and exhibits impressive prediction-related performance without requiring human-engineered features. Examples include the recency effect [5] and the contextualized trial sequence [22]. In teh DKT models, a sequence of hidden states $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n)$ is computed to encode the sequential information obtained from previous interactions. In each timestep t, the model calculates the hidden state $\boldsymbol{h_t}$ and the student's response p_t as follows:

$$\begin{aligned} \boldsymbol{h_t} &= \operatorname{Tanh}\left(\boldsymbol{W_{hx}} \boldsymbol{x_t} + \boldsymbol{W_{hh}} \boldsymbol{h_{t-1}} + b_h\right) \\ p_t &= \sigma\left(\boldsymbol{W_{hy}} \boldsymbol{h_t} + b_p\right) \end{aligned} . \tag{1}$$

However, the latent encoding of the knowledge state in the DKT cannot consistently depict the students' mastery of the KCs and predict temporal changes in the knowledge state over time. To solve the major problems of the DKT in terms of modeling the KC, the DKT+ model introduces regularization terms, which correspond to reconstruction and waviness, to the loss function of the original DKT model to enhance the consistency of predictions. The results of experiments have shown that the regularized loss function can solve the above two problems without degrading the performance of the DKT on the original task [43]. Chen et al. [7] sought to solve the problem of data sparsity by incorporating prerequisite concept pairs as constraints into the DKT model, and to improve predictions of the students' mastery of concepts while offering a partial interpretation of the results. Wang et al. [42] introduced HawkesKT, a novel approach in KT that leverages the point process to dynamically model temporal cross-effects. Recognizing that a student's proficiency in a **knowledge component** (**KC**) is influenced not only by their own past interactions with that KC but also by interactions with other KCs, the model captures these intricate cross-effects. Crucially, it acknowledges the varying temporal evolutions of these effects on different

KCs. This model marks a significant step in KT, offering a more nuanced understanding of how students' knowledge evolves over time. However, despite these advantages, the hidden state variables of a neural network cannot explicitly represent explainable educational meanings without inducing a prior cognitive structure and the attendant constraints. It remains a critical challenge to accurately characterize changes in the students' knowledge states in order to support accurate predictions in complex and diverse question-answering scenarios by using a DNN model.

2.2.2 Deep Knowledge Tracing with Memory Augmentation. To enhance the capacity of storage of the knowledge state, memory-aware KT, which is inspired by memory-augmented neural networks [14], introduces an external memory module to store the knowledge states and update the corresponding mastery of knowledge by the student. The **dynamic key—value memory network** (**DKVMN**) [48] initializes a static matrix called a key matrix to store latent KCs, and a dynamic matrix called a value matrix to store and update the mastery of the corresponding KCs through read and write operations over time. The DKVMN defines a static key matrix to represent concepts and a dynamic value matrix to track the state of the students' concepts in the framework of the **memory-augmented neural network** (**MANN**). The DKVMN can automatically discover the concepts underlying exercises, where this is typically obtained through manual annotations, and can depict the changing knowledge state of a student. In addition, the **sequential key—value memory network** (**SKVMN**) [1] can address the limitation in the DKT and the DKVMN whereby the KCs required to answer questions in past exercises in a sequence are not necessarily relevant to the KCs required to answer questions in the current exercise. A modified LSTM called Hop-LSTM is used in the SKVMN to hop across LSTM cells according to the relevance of the latent KCs, where this can be used to directly capture long-term dependencies. When calculating the growth in knowledge owing to a new exercise in the writing process, the SKVMN enables it to consider the current state of knowledge in order to obtain accurate results. Furthermore, Deep-IRT is a synthesis of the model of **item response theory** (**IRT**) and a KT model based on the DKVMN to render deep learning-based KT explainable. Specifically, the DKVMN model is used to process the student's learning trajectory and estimate the level of difficulty of the item as well as the student's ability over time. Then, the IRT model is used to estimate the probability that a student will answer an item correctly by using the estimated knowledge state and the difficulty of the given item.

2.2.3 Deep Knowledge Tracing with Attention Mechanism. The attention mechanism [26, 33, 39, 44–47] is effective on tasks involving sequence modeling. The idea underlying this mechanism is to focus on the relevant elements of the input signals when predicting the output. **Self-attentive knowledge tracing** (**SAKT**) [31] is the first method to use attention mechanisms in the context of KT. Attention mechanisms are more flexible than recurrent and memory-based neural networks. The results of extensive experiments on a variety of real-world datasets suggest that the SAKT model can outperform state-of-the-art methods, and is one order of magnitude faster than RNN-based approaches. Pandey and Srivastava [32] also introduced a novel Relation-aware self-attention model for Knowledge Tracing (RKT), which integrates exercise relation information, student performance data, and student forget behavior into the contextual information, showing improved performance on three real-world datasets and providing interpretable attention weights for visualizing the relation between interactions and temporal patterns in the learning process. Ghosh et al. [11] proposed a context-aware AKT model by incorporating the self-attention mechanism into cognitive and psychometric models. They defined context-aware representations of exercises and responses by using a monotonic attention mechanism to summarize every learner's historical performance on an appropriate time scale. They used the Rasch model to capture differences between exercises covering the same concept. Separated self-attentive neural knowledge tracing (SAINT), proposed by Choi et al. [8], has an encoder—decoder structure in which the sequences of embeddings of the exercise and the response are entered into the encoder and the decoder, respectively. The encoder applies the self-attention layers to the sequence of exercise embeddings, and the decoder alternately applies self-attention layers and encoder—decoder attention layers to the sequence of response embeddings. This separation of the inputs allows the model to stack attention layers multiple times, resulting in an improvement in the **area under the receiver-operating characteristic curve** (**AUC**). This study is the first to propose an encoder–decoder model for KT that applies deep self-attentive layers separately to the exercises and the responses. Based on SAINT, SAINT+ [35] has an encoder-decoder structure in which the encoder applies self-attention layers to a stream of exercise embeddings, and the decoder alternately applies self-attention layers and encoder—decoder attention layers to streams of response embeddings and the output of the encoder. Moreover, SAINT+ incorporates two temporal feature embeddings into its response embeddings: the elapsed time, the time taken for a student to answer a question, and the lag time, the interval between adjacent learning activities. He et al. [18] addresses the need for enhanced **representation learning** (**RL**) in **Deep Learning-based Knowledge Tracing** (**DLKT**) methods, an area often overlooked in favor of model structure innovations. It explores four key types of factors: exercise and skill attributes, learners' historical performance, and their forgetting behavior. This work highlights the critical role of nuanced RL in advancing the effectiveness of DLKT methods. He et al. [17] also presented the **Memory-augmented Attentive Network** (**MAN**) to tackle the **Skill Switching Phenomenon** (**SSP**) in KT, utilizing advanced deep learning techniques. MAN employs memory-augmented neural networks for encoding long-term memory knowledge and attention-based networks to process recent knowledge. The integration of a novel context-aware attention mechanism effectively harmonizes these two types of knowledge. This innovative approach significantly enhances the management of SSP in KT. Cui et al. [10] introduced a novel model, MRT-KT, that enables fine-grained interaction modeling between question-response pairs by implementing a unique relation encoding scheme based on KCs and student performance. However, none of the prevent methods can overcome issues related to predictive accuracy that are caused by inadequate capture of the knowledge state in complex scenarios. Moreover, the problem of noise induced by anomalous interaction has not been adequately considered and solved.

3 PRELIMINARIES

In this section, we provide preliminariy information on the encoder module and the causal convolution module, which are two standard components frequently used in the proposed ELAKT model.

3.1 Encoder Module

The encoder module is an essential sub-module in the well-known transformer model. Assume $\boldsymbol{Q} \in \mathbb{R}^{T\times D_m}$, $\boldsymbol{K} \in \mathbb{R}^{T\times D_m}$, and $\boldsymbol{V} \in \mathbb{R}^{T\times D_m}$, with each representing the encoder's input of queries, keys, and values, respectively. When there are D_h heads, the encoder module projects each $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ to a latent space by multiplying the matrices $\boldsymbol{W_i^Q} \in \mathbb{R}^{D_m\times D_m/D_h}$, $\boldsymbol{W_i^K} \in \mathbb{R}^{D_m\times D_m/D_h}$, and $\boldsymbol{W_i^V} \in \mathbb{R}^{D_m\times D_m/D_h}$, that is,

$$\boldsymbol{Q_i} = \boldsymbol{QW_i^Q}, \boldsymbol{K_i} = \boldsymbol{KW_i^K}, \boldsymbol{V_i} = \boldsymbol{VW_i^V}. \tag{2}$$

The obtained $\boldsymbol{Q_i}$, $\boldsymbol{K_i}$, and $\boldsymbol{V_i}$ are then fed forward to a masked multi-head attention layer.

3.1.1 *Masked Multi-head Attention Layer.*

Because the keys and values after the current position include the information to be predicted, a masking mechanism is used in the masked multi-head attention layer, whereby only the lower-triangular portion of the scaled dot product [39] matrix obtained from $\boldsymbol{Q_i K_i^T}$ is used to calculate the attention heads. Each attention head is calculated by:

$$head_i = \text{Softmax}\left(\text{Mask}\left(\frac{\boldsymbol{Q_i K_i^T}}{\sqrt{D_m}}\right)\right)\boldsymbol{V_i}, \tag{3}$$

Then, a concatenation of D_h attention heads ($D_h = 4$ by default in the ELAKT model) is multiplied by $W^O \in \mathbb{R}^{D_m \times D_m}$ to aggregate all the obtained attention heads. This concatenated tensor is the output of the masked multi-head attention layer, that is,

$$\text{MultiHead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right) = \text{Concat}\left(head_1, \cdots, head_{D_h}\right) W^O. \tag{4}$$

3.1.2 *Position-wise Feed-forward Layer.* The position-wise feed-forward layer is also known as the position-wise **feed-forward network** (**FFN**). It is a fully-connected network that operates separately and identically at each position:

$$\text{FFN}(\boldsymbol{M}) = \text{ReLU}\left(\boldsymbol{M}\boldsymbol{W}^1 + \boldsymbol{b}^1\right)\boldsymbol{W}^2 + \boldsymbol{b}^2, \tag{5}$$

where $\boldsymbol{M} = \text{Multihead}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right)$. $\{\boldsymbol{W}^1 \in \mathbb{R}^{D_m \times (D_m * z)}, \boldsymbol{W}^2 \in \mathbb{R}^{(D_m * z) \times D_m}\}$ and $\{\boldsymbol{b}^1 \in \mathbb{R}^{D_m * z}, \boldsymbol{b}^2 \in \mathbb{R}^{D_m}\}$ are trainable weight matrices and bias vectors, respectively. Typically, the intermediate dimension $(D_m * z)$ of the FFN is set to be larger than D_m [27]. In ELAKT model, the value of z is set to four by default.

3.1.3 *Residual Connection and Normalization.* The encoder module uses the residual connection [16] and the layer normalization [4] techniques. Residual connection is a commonly used structural technique in the transformer model that can help avoid the problem of vanishing or exploding gradients while improving the speed of convergence of the model. Given the definitions of the aforementioned two layers, the encoder module can be written as:

$$\begin{aligned} \boldsymbol{M} &= \boldsymbol{V} + \text{Dropout}\left(\text{Multihead}\left(\text{LayerNorm}\left(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\right)\right)\right) \\ \boldsymbol{H} &= M + \text{Dropout}(\text{FFN}(\text{LayerNorm}(\boldsymbol{M}))) \end{aligned}. \tag{6}$$

$\boldsymbol{H} \in \mathbb{R}^{T \times D_m}$ is the output of encoder module, where $\boldsymbol{h_t}$ is the tth row of $\boldsymbol{H}$. The Multihead($\cdot$) denotes the masked multi-head attention layer defined in 3.1.1 and LayerNorm($\cdot$) denotes the layer normalization operation. The operations in (6) constitute the content of an encoder layer:

$$\boldsymbol{H} = \text{Encoder}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}). \tag{7}$$

Note that sequentially aligned multiple copies of encoder layers constitute the encoder module. By default, the number of copies is set to one in this article.

3.2 Causal Convolution Module

The causal Convolution [30] is a type of convolution operation in deep learning, particularly in sequence-to-sequence models. For the special case of the standard 1D convolution, the causal convolution can be easily implemented by applying left zero-padding to the input of a normal convolution. Mathematically, given a sequence input $\boldsymbol{Y} = (\boldsymbol{y_1}, \ldots, \boldsymbol{y_t}, \ldots, \boldsymbol{y_T}), \boldsymbol{y_t} \in \mathbb{R}^{D_m}, \boldsymbol{Y} \in \mathbb{R}^{D_m \times T}$, and $\boldsymbol{W} \in \mathbb{R}^{D_m \times D_m \times K}$ is the convolution kernel (trainable weight matrices), the causal convolution at timestep t is represented as

$$\begin{aligned} \boldsymbol{c_t} &= (c_t^1, c_t^2, \ldots, c_t^{D_m}) = \text{CausalConv1d}(Y, kernel\_size = K) \\ c_t^i &= \boldsymbol{Y} \otimes \boldsymbol{W_i} = \sum_{j=0}^{D_m - 1} \sum_{k=0}^{K-1} \boldsymbol{W_{i,j,k}} \cdot Y_{\boldsymbol{D_m - j, t - k}} \end{aligned}, \tag{8}$$

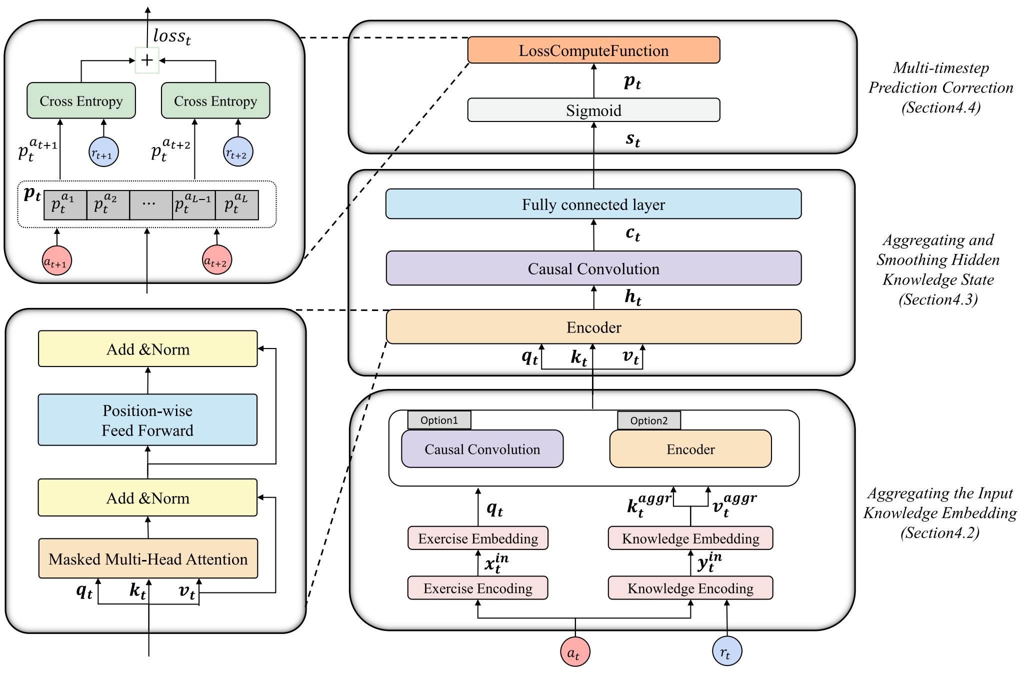


Fig. 2. The overall structure of the ELAKT model consisting three modules: those for aggregating the input knowledge embedding, aggregating and smoothing hidden knowledge state, and multi-timestep prediction correction.

where $\boldsymbol{c}_t \in \mathbb{R}^{D_m}$, K is the kernel size, and D_m is the number of input and output channels. The causal convolution is a type of time-limited convolution that is commonly used for processing sequences in the time sequence domain, such as in speech recognition and audio processing [24, 41]. Thus, the output $\boldsymbol{c}_t$ of the causal convolution does not depend on any of the future input timesteps $(\boldsymbol{y}_{t+1}, \boldsymbol{y}_{t+2}, \ldots, \boldsymbol{y}_T)$. Recently, several researchers have explored the use of the causal convolution as a module for aggregating contextual information [15] and contextual smoothing [49] in the domain of time sequence. These approaches have yielded promising improvements in performance on the respective tasks. We use the causal convolution module for two distinct purposes: 1) aggregating the input knowledge embeddings (Section 4.2), and 2) aggregating and smoothing the knowledge states (Section 4.3).

4 METHODOLOGY

In this section, we first present the problem formulation and then introduce the proposed ELAKT method, which is based on the attentive KT model. An overview of the architecture of ELAKT is illustrated in Figure 2. It consists of three major modules: those for aggregating the input knowledge embedding, aggregating and smoothing the hidden knowledge states, and correcting the predictions over multiple timesteps. In this architecture, the first encoder layer and causal convolution layer in the bottom module are two interchangeable options for aggregating the input knowledge embeddings. The second encoder layer in the middle module is designed to capture temporal dependencies and leverage the attention mechanism. The second causal convolution layer stacked on top of it is designed to aggregate and smooth the student's hidden knowledge states to improve the attention of the contextual knowledge state. This causal convolution layer can help tackle the presence of stochastic behaviors, that is, anomalies in the knowledge state.

4.1 Problem Formulation

Let $Q = \{a_1, a_2, \ldots, a_M\}$ be the set of all exercises and M be the number of exercises, where $a_t \in \mathbb{N}^+$ is the index of the exercise at timestep t. Let $\mathcal{E} = \{c_1, c_2, \ldots, c_L\}$ be the set of all KCs and L be the number of KCs, where $c_t \in \mathbb{N}^+$ is the index of the exercise at timestep t. Let $r_t \in \{0, 1\}$ be the correctness of the student's answer to exercise a_t at time t, where 0 represents an incorrect response and 1 represents the correct one. A KT problem can be defined as follows: *Given a student's past interactions with exercises $\mathcal{Z} = \{(a_1, r_1), \ldots, (a_t, r_t)\}$, predict the probabilities of the student's responses in the upcoming j timesteps, that is,* $\{P(r_{t+i} = 1 \mid \mathcal{Z}) | i = 1, 2, \ldots, j\}$. Note that in most real-world educational settings, the size of the exercise set is considerably larger than the set of concepts invoked by it, and many exercises are assigned to only a few students. Therefore, using the index of the exercise as the input yields sparse knowledge embeddings. To avoid this issue, a majority of current KT methods [11, 31] use concepts to assign indices to exercises, and all exercises covering the same concept are treated as a single exercise. In this case, $c_t = a_t$ and $M = L$. We make use of this setting as well. The mathematical notation used in this article is summarized in Table 1. The first part includes scalars, the second includes vectors, and the third includes sets.

4.2 Aggregating the Input Knowledge Embedding

At timestep t, the model receives the input (a_t, r_t). We define the following *representation of exercise encoding* $\boldsymbol{x}_t^{in}$ through the one-hot encoding of exercise a_t:

$$\boldsymbol{x}_t^{in} = (o_0, o_1, \ldots, o_L); \quad \boldsymbol{x}_t^{in} \in \{0, 1\}^L, \tag{9}$$

where L is the total number of concepts, $o_{a_t} = 1$, and the remaining terms are zero. We also define the *representation of knowledge encoding* $\boldsymbol{y}_t^{in} \in \{0, 1\}^{2L}$ based on the representation of exercise encoding $\boldsymbol{x}_t^{in}$ and the response r_t to it:

$$\boldsymbol{y}_t^{in} = \begin{cases} \left[\boldsymbol{x}_t^{in} \oplus \boldsymbol{0}\right] & \text{if} \quad r_t = 1 \\ \left[\boldsymbol{0} \oplus \boldsymbol{x}_t^{in}\right] & \text{if} \quad r_t = 0 \end{cases}, \tag{10}$$

where $\boldsymbol{0} = (0, 0, \ldots, 0)$ with the length of L, and $\oplus$ is the operation that concatenates two vectors. For the same exercise, the knowledge states of correct and incorrect responses should be distinguishable. To this end, in our design, if the non-zero element appears in the first half of the vector, it implies a correct response with the location of the non-zero element indicating which exercise is correctly answered. Similarly, the appearance of non-zero elements in the second half of the vector implies an incorrect response.

The exercise encoding $\boldsymbol{x}_t^{in}$ is fed to a linear **fully-connected** (**FC**) layer to generate the exercise embedding $\boldsymbol{x}_t \in \mathbb{R}^{D_m}$, which is treated as the query $\boldsymbol{q}_t$ in the ELAKT model. The knowledge encoding $\boldsymbol{y}_t^{in}$ is fed to another FC layer to generate the knowledge embedding $\boldsymbol{y}_t \in \mathbb{R}^{D_m}$, which is treated as both the aggregated keys $\boldsymbol{k}_t^{\text{aggr}}$ and the aggregated values $\boldsymbol{v}_t^{\text{aggr}}$. Specifically, we have

$$\begin{aligned} \boldsymbol{q}_t &= \boldsymbol{x}_t = \text{FC}(\boldsymbol{x}_t^{in}; \text{dim\_in} = L, \text{dim\_out} = D_m); \\ \boldsymbol{k}_t^{\text{aggr}} = \boldsymbol{v}_t^{\text{aggr}} &= \boldsymbol{y}_t = \text{FC}(\boldsymbol{y}_t^{in}; \text{dim\_in} = 2L, \text{dim\_out} = D_m). \end{aligned} \tag{11}$$

In the self-attention layers of a traditional transformer, the similarities between queries and keys are computed based on their point-wise values. A common practice is to use the embeddings of concept IDs as queries, but this leads to poor attention to exercises involving different concepts, making it difficult to capture the comprehensive knowledge state. In this section, we introduce two methods to aggregate the input knowledge: the causal convolution, and the encoder module. Figure 3 illustrates details of the processing of the input data by using these two methods.

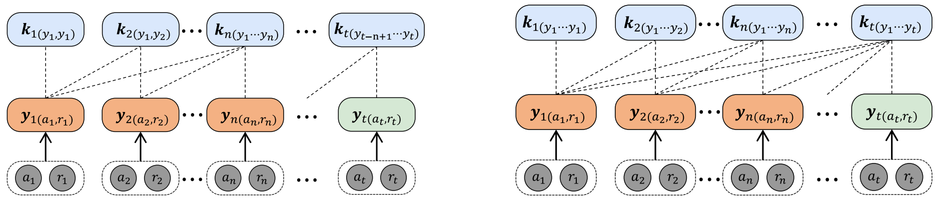
Table 1. Notations and Descriptions

Notation	Description
t	timestep
a_t	index of exercise (or item) in timestep t
c_t	index of KC in timestep t
r_t	response correctness in timestep t
$s_t^{c_i}$	the state of knowledge of concept c_i in timestep t
$p_t^{a_{t+1}}$	the probability of correctly responding to exercise a_{t+1} based on the estimated probability $\boldsymbol{p}_t$ in timestep t
β	decay coefficient in the module “multi-timestep prediction correction” with default setting of 0.5
u	predicted response length in module “multi-timestep prediction correction” with default setting of 2
T	length of sequence of the student’s responses with default setting of 80
M	number of exercises
N	number of students
L	number of concepts
D_m	dimensionality of the ELAKT model’s hidden states with default setting of 128
n	kernel size of causal convolution in module “input knowledge embedding aggregation”
e	kernel size of causal convolution in module “aggregating and smoothing hidden knowledge state”
$\boldsymbol{x}$	representation of exercise embedding
$\boldsymbol{y}$	representation of knowledge embedding
$\boldsymbol{p}$	probability of correctly answering the exercise
$\boldsymbol{q}$	query of the multi-head attention in the module “hidden knowledge state aggregation”
$\boldsymbol{k}, \boldsymbol{v}$	key and value of the multi-head attention in the module “hidden knowledge state aggregation”
$\boldsymbol{k}^{\text{aggr}}, \boldsymbol{v}^{\text{aggr}}$	key and value of the multi-head attention in the module “input knowledge embedding aggregation”
$\boldsymbol{c}$	student’s knowledge state output by the causal convolution
$\boldsymbol{h}$	student’s hidden knowledge state
$\boldsymbol{s}$	student’s full state of KC
$\mathcal{Z}$	the set of exercise-based interactions of a single student
$\mathcal{Q}$	the set of all exercises
$\mathcal{E}$	the set of all concepts

1) Causal Convolution. We use the causal convolution of kernel size n to transform the inputs into values and keys (note that the query remains unchanged and the exercise embedding $\boldsymbol{x}_t$) is used:

$$\boldsymbol{k}_t = \boldsymbol{v}_t = \text{CausalConv1d}(\boldsymbol{y}_{t-n+1}, \ldots, \boldsymbol{y}_{t-1}, \boldsymbol{y}_t; kernel\_size = n). \tag{12}$$

Equation (8) describes how the causal convolution generates a sequence by aggregating the input embedding of contextual knowledge. Specifically, the convolution kernel $\boldsymbol{W}$ performs a weighted sum of the current timestep and its previous timesteps in the input sequence. This weighted sum can be considered to be an aggregation of the contextual information.



(a) Aggregation using causal convolution (b) Aggregation using the encoder module

Fig. 3. Detailed structures of two methods to aggregate input embeddings.

Thus, the generated key $\boldsymbol{k}_t$ and value $\boldsymbol{v}_t$ contain more comprehensive knowledge such that the similarity can be appropriately calculated according to information on the implicit KC of the exercises. At the same time, the causal convolution ensures that $\boldsymbol{k}_t$ and $\boldsymbol{v}_t$ can obtain information from only $\boldsymbol{y}_1$ to $\boldsymbol{y}_t$, thus guaranteeing that the temporal order is not violated in the exercise sequence.

2) Encoder. We also introduce another option, that is, the encoder in Equation (7), as an alternative for aggregating contextual knowledge embeddings. The process can be expressed as follows:

$$\begin{aligned} \boldsymbol{Q}^{\text{aggr}} &= (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_t, \ldots, \boldsymbol{q}_T); \quad \boldsymbol{Q}^{\text{aggr}} \in \mathbb{R}^{T\times D_m} \\ \boldsymbol{K}^{\text{aggr}} &= (\boldsymbol{k}_1^{\text{aggr}}, \ldots, \boldsymbol{k}_t^{\text{aggr}}, \ldots, \boldsymbol{k}_T^{\text{aggr}}); \quad \boldsymbol{K}^{\text{aggr}} \in \mathbb{R}^{T\times D_m} \\ \boldsymbol{V}^{\text{aggr}} &= (\boldsymbol{v}_1^{\text{aggr}}, \ldots, \boldsymbol{v}_t^{\text{aggr}}, \ldots, \boldsymbol{v}_T^{\text{aggr}}); \quad \boldsymbol{V}^{\text{aggr}} \in \mathbb{R}^{T\times D_m} \\ \boldsymbol{R} &= \text{Encoder}(\boldsymbol{Q}^{\text{aggr}}, \boldsymbol{K}^{\text{aggr}}, \boldsymbol{V}^{\text{aggr}}); \quad \boldsymbol{R} \in \mathbb{R}^{T\times D_m} \end{aligned}. \tag{13}$$

The core component of the encoder is the self-attention mechanism. It allows the model to weigh the importance of each input in relation to every other input in the sequence. In each timestep, the masked self-attention mechanism computes attention scores as the output based on the previous input knowledge embedding. In this way, the output representation incorporates information from other input knowledge embeddings.

Let $\boldsymbol{k}_t = \boldsymbol{v}_t = \boldsymbol{R}_t$. Thus far, we have obtained $\boldsymbol{q}_t$, $\boldsymbol{k}_t$, and $\boldsymbol{v}_t$ by using either of the two methods introduced above. Note that both the causal convolution and masked self-attention in the encoder can aggregate contextual information while preserving its temporal order. However, the methods differ in their ability to capture dependencies with a flexible range: The receptive field of the causal convolution grows linearly with the number of layers, and depends on the size of hyper-parameters of the kernel, while masked self-attention in the encoder can capture dependencies with a flexible range regardless of the size of the kernel or the number of layers.

4.3 Aggregating and Smoothing Hidden Knowledge State

The $\boldsymbol{q}_t$, $\boldsymbol{k}_t$, and $\boldsymbol{v}_t$ obtained in the previous step are inputted to another encoder. The encoder module takes $\boldsymbol{Q}_{in} \in \mathbb{R}^{T\times D_m}$, $\boldsymbol{K}_{in} \in \mathbb{R}^{T\times D_m}$, and $\boldsymbol{V}_{in} \in \mathbb{R}^{T\times D_m}$ to represent the sequences of queries, keys, and values, respectively:

$$\begin{aligned} \boldsymbol{Q}_{in} &= (\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_T); \\ \boldsymbol{K}_{in} &= (\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_T); \\ \boldsymbol{V}_{in} &= (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T); \\ \boldsymbol{H} &= \text{Encoder}(\boldsymbol{Q}_{in}, \boldsymbol{K}_{in}, \boldsymbol{V}_{in}) = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T); \quad \boldsymbol{H} \in \mathbb{R}^{T\times D_m}, \end{aligned} \tag{14}$$

where $\boldsymbol{h_t} \in \mathbb{R}^{D_m}$ is the t-th row of $\boldsymbol{H}$. Traditional attentive KT methods predominantly consider $\boldsymbol{h_t}$ as the knowledge state at timestep t to predict the response at the next timestep based on $\boldsymbol{h_t}$. However, when the local exercise a_t is altered from invoking a KC c_1 to a new one c_2, this transition may cause the student's state of hidden knowledge $\boldsymbol{h_t}$ to fluctuate, where this is inconsistent with the actual learning process. Furthermore, the presence of stochastic behaviors leads to anomalies in the knowledge state. Therefore, we introduce a layer of causal convolution after the encoder to improve the attention of the contextual knowledge state. This process can be formulated as follows:

$$\boldsymbol{c_t} = \text{CausalConv1d}\left(\boldsymbol{h_{t-e+1}}, \ldots, \boldsymbol{h_{t-1}}, \boldsymbol{h_t}; kernel\_size = e\right); \quad \boldsymbol{c_t} \in \mathbb{R}^{D_m}. \tag{15}$$

The causal convolution incorporates the knowledge states $\boldsymbol{h_{t-e+1}}, \ldots, \boldsymbol{h_{t-1}}$ adjacent to $\boldsymbol{h_t}$. Assuming that a student's knowledge state does not change drastically between adjacent timesteps, the weighted summation of the causal convolution can be considered to be a mechanism to smooth the knowledge state. Furthermore, the number of convolution kernels in the causal convolution depends on the number of output channels. During the convolution operation, the input sequence is convolved with the corresponding convolution kernel $\boldsymbol{W_i}$ to obtain the corresponding results of the output channel c_t^i. This setting is used to learn and extract rich features and patterns to aggregate multiple components of the knowledge state in the input knowledge state. The "kernel_size" hyper-parameter, that is, e, can be tuned to improve predictive accuracy. We will comprehensively discuss the impact of the hyper-parameter e on the predictive performance of the model in our ablation experiment later in this article. Traditional attentive KT methods map $\boldsymbol{h_t}$ to a **one-dimensional (1D)** scalar through a fully-connected layer to predict the probability of a student responding to the exercise. The ELAKT model uses knowledge aggregation and the causal convolution of knowledge states such that the matrix $\boldsymbol{c_t}$ contains the comprehensive knowledge state.

We then add a fully-connected layer to change the number of dimensions $\boldsymbol{c_t}$ from D_m to L (the number of concepts). This process can be expressed as follows:

$$\boldsymbol{s_t} = \text{FC}(\boldsymbol{c_t}; dim\_in = D_m, dim\_out = L); \quad \boldsymbol{s_t} \in \mathbb{R}^L. \tag{16}$$

The $\boldsymbol{s_t}$, which is obtained through Equation (15) and Equation (16), contains knowledge states with a one-to-one mapping to the KCs, and thus can predict responses at multiple timesteps in the future. This capability relies upon the multi-timestep prediction correction module presented in the next section.

4.4 Multi-timestep Prediction Correction

A student may perform guessing and slipping when answering exercises, that is, the stochastic behaviors. The student may give correct responses by guessing answers to the exercises even if the relevant KCs have not been mastered. To solve this problem, we propose the multi-timestep prediction module based on the assumption that the student's knowledge state should exhibit steady and smooth changes during a short period of time (i.e., over multiple subsequent exercises). The core idea is that we can make use of a student's current knowledge state to predict the probability of the student's responses to exercises over multiple timesteps in the future. We can reduce the error in the inferences of the knowledge states by adding loss to the upcoming timesteps. To this end, we first obtain the response probability $\boldsymbol{p_t}$ by applying the sigmoid activation layer to $\boldsymbol{s_t}$, and then use it to predict the response-related performance of the student:

$$\boldsymbol{p_t} = \text{Sigmod}\left(\boldsymbol{s_t}\right) = \left(p_t^{a_1}, p_t^{a_2}, \ldots, p_t^{a_L}\right); \quad \boldsymbol{p_t} \in [0, 1]^L, \tag{17}$$

where $p_t^{a_{t+1}} \in [0,1]$ is the probability of predicting the student's response to exercise a_{t+1} at timestep $t+1$. Accordingly, the loss function at timestep t can be defined as the following cross-entropy:

$$loss_t = -\sum_{i=1}^{u} \beta^i [r_{t+i} \log\left(p_t^{a_{t+i}}\right) + (1 - r_{t+i}) \log\left(1 - p_t^{a_{t+i}}\right)], \tag{18}$$

where $\beta \in (0,1)$ represents the coefficient of decay and u denotes the length of the future timesteps to be predicted. By default, we set these two parameters to 0.5 and 2, respectively.

The multi-timestep prediction correction module can be treated as a form of regularization to bring about enhanced generalization: Predicting multiple timesteps and calculating the loss in comparison with the ground truth can regularize the training to prevent overfitting and enable the model to learn more robust features. As a result, the post-trained model is able to generalize beyond the immediate interactions to make accurate predictions in situations where the input data are noisy or incomplete. Based on the virtue, the multi-timestep prediction correction module can handle the noise introduced by stochastic behaviors, such as guessing and slipping.

4.5 Description of the Algorithm

The pseudo-code of the ELAKT framework is presented in Algorithm 1. Lines 4–6 initialize the exercise embedding $\boldsymbol{x_t}$ and the knowledge embedding $\boldsymbol{y_t}$ from an input data point (a_t, r_t). The one-hot encodings $\boldsymbol{x}_t^{in}$ and $\boldsymbol{y}_t^{in}$ are first generated by using a_t and (a_t, r_t), respectively. Then, a fully-connected layer is used to map the dimensions of $\boldsymbol{x}_t^{in}$ and $\boldsymbol{y}_t^{in}$ to the input dimensions of the model D_m to obtain $\boldsymbol{x_t}$ and $\boldsymbol{y_t}$. The algorithm uses two methods for aggregating input knowledge to generate $\boldsymbol{q_t}$, $\boldsymbol{k_t}$, and $\boldsymbol{v_t}$. When *aggrmethod* is set to one, the algorithm uses causal convolution (Lines 7—9) to aggregate the input knowledge embeddings by Equation (12), whereas, when *aggrmethod* is set to two, the algorithm uses the encoder module (Lines 10—14) to aggregate the input knowledge embeddings by Equation (13). The hidden knowledge state $\boldsymbol{h_t}$ is generated from $\boldsymbol{q_t}$, $\boldsymbol{k_t}$, and $\boldsymbol{v_t}$ via the encoder model (Lines 15—16) by Equation (14), and is further fed into the causal convolution for aggregation and smoothing by Equation (15) (Line 17). This yields $\boldsymbol{c_t}$. In Line 18, a fully-connected layer is used to map the dimensions of $\boldsymbol{c_t}$ from D_m to L, thus yielding the state of the KC $\boldsymbol{s_t}$ by Equation (16). The vector $\boldsymbol{p_t}$, representing the response probabilities of the KCs, is obtained by using Equation (17) (Line 19). Finally, in Line 20, the loss over multiple timesteps is computed by using Equation (18). In Line 22, the loss obtained from T timesteps for each student iterated in Line 3 is summed to obtain the total loss $\mathcal{T}_n$ for all interactions of a single student. Finally, the loss incurred in all interactions by all students is summed up to calculate the final loss $\mathcal{L}$. The ELAKT model is trained based on this. Lines 4—18 of Algorithm 1 correspond to the claimed contribution 1 and the contribution 2 for solving the first challenge in AKT. Lines 19—21 of Algorithm 1 correspond to the claimed contribution 3 to solve the second challenge.

5 IMPLEMENTATION AND EXPERIMENTAL RESULTS

5.1 Details of Implementation

We now specify details of the implementation of the ELAKT model. The model under the attention mechanism had 128 dimensions, and the maximum allowed sequence length l was 80. The value of kernel_size of CausalCov1d in the modules to aggregate input knowledge embeddings, and to aggregate and smooth hidden knowledge states were set to five by default. There were four attention heads. We used the Adam optimizer with a learning rate of 0.001. The dropout rate was set to 0.1 to deal with overfitting and the L2 weight decay was set to 10^{-6}. All the model parameters were initialized to zero unless otherwise specified. All the experiments were conducted on Tesla A100 PCIe 40 GB GPUs, with a 2 * Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz, 64 GB DDR4

ALGORITHM 1: The Pseudocode of the ELAKT Framework

```
Input: Student historical response dataset $\mathbb{D} = \{S_1, S_2, \ldots, S_N\}$, where $S_n = \{(a_1, r_1), \ldots, (a_T, r_T)\}$,
       $N$: student number, $T$: length of sequence of student response, $L$: number of exercises,
       $aggrmethod \in \{1, 2\}$: parameters of the aggregation method
Output: Student response probability set, $P_{\mathbb{D}} = \{P_{S_1}, \ldots, P_{S_N}\}$, where $P_{S_n} = \{p_1^{a_2}, p_2^{a_3}, \ldots, p_T^{a_T}\}$
1  repeat
2      foreach $S_n$ in $\mathbb{D}$ do
3          foreach $(a_t, r_t) \in S_n$ do
               /* Implement the Aggregation of Input Knowledge Embedding (in
                  Section 4.2) */
4              $x_t^{in} \leftarrow$ ExerciseEncoding($a_t$)    // Exercise Encoding(Equation (9))
5              $y_t^{in} \leftarrow$ KnowledgeEncoding($a_t, r_t$)    // Knowledge Encoding(Equation (10))
               $x_t \leftarrow FC(x_t^{in})$
6              $y_t \leftarrow FC(y_t^{in})$ // Convert input encoding to input embedding(Equation (11))
7              if $aggrmethod = 1$ then
8                  $q_t \leftarrow x_t$
9                  $k_t, v_t \leftarrow$ CausalConv1d($y_{t-n+1}, \ldots, y_t$) // Aggregation w. causal convolution
                    (Equation (12))
10             else if $aggrmethod = 2$ then
11                 $q_t \leftarrow x_t$
12                 $k_t^{\text{aggr}}, v_t^{\text{aggr}} \leftarrow y_t, y_t$
13                 $Q^{\text{aggr}}, K^{\text{aggr}}, V^{\text{aggr}} \leftarrow q_t, k_t^{\text{aggr}}, v_t^{\text{aggr}}$
14                 $k_t, v_t \leftarrow$ Encoder($Q^{\text{aggr}}, K^{\text{aggr}}, V^{\text{aggr}}$)    // Aggregation with the encoder
                    module (Equation (13))
               /* Aggregating and Smoothing Hidden Knowledge State (in Section 4.3) */
15             $Q_{in}, K_{in}, V_{in} \leftarrow q_t, k_t, v_t$
16             $h_t \leftarrow$ Encoder($Q_{in}, K_{in}, V_{in}$)    // Generate hidden knowledge state
                (Equation (14))
17             $c_t \leftarrow$ CausalConv1d($h_{t-e+1}, \ldots, h_t$)    // hidden knowledge state aggregation
                (Equation (15))
18             $s_t \leftarrow FC(c_t)$    // Generate knowledge concept state (Equation (16))
               /* Multi-timestep Prediction Correction (in Section 4.4) */
19             $p_t = (p_t^{a_1}, \ldots, p_t^{a_L}) \leftarrow$ Sigmod($s_t$)    // Calculate probability of response
                knowledge concept (Equation (17))
20             $p_t^{a_{t+1}}, p_t^{a_{t+2}}, \ldots, p_t^{a_{t+u}} \leftarrow p_t$    // Calculate the response probability for
                future exercises
21             $loss_t \leftarrow -\sum_{i=1}^{u}[\beta^i * \text{crossEntropy}(p_t^{a_{t+i}}, r_{t+i})]$    // Calculate loss in multiple
                timesteps (Equation (18))
22         $\mathcal{T}_n = \Sigma_{t=1}^{T} loss_t$
23     $\mathcal{L} = \Sigma_{n=1}^{N} \mathcal{T}_n$
24     update learning variables using Adam optimizer on $\mathcal{L}$
25 until $\mathcal{L}$ reaches the convergence condition
```

RAM, and 1* 256 GB SATA SSD. The software environment was Python 3.9.12, NVIDIA Driver Version 525.78.01, and PyTorch 1.12.1+cu116.

5.2 Method of Evaluation

5.2.1 Baseline Models and Evaluation Metrics. We compared the proposed ELAKT against several baseline KT methods, including DKT, SAKT, AKT, and DKVMN. The prediction task was

Table 2. Overview of the Experimental Datasets

Dataset	Student	Concept	Interaction	Average length	Length variance	Maximum length
ASSISTments09-10	4,151	110	325,637	78	24,293.4	1,261
ASSISTments2015	19,917	100	708,631	35	2,542.6	632
ASSISTments2017	1,709	102	942,816	551	175,529.1	3,057
EdNet1-Sub	10,000	108	2,706,954	271	945,683.0	26,255

considered to be a binary classification problem, that is, the aim was to predict whether an exercise was answered correctly. We quantified predictive performance by using the AUC. We also used the accuracy (ACC), **mean absolute error** (**MAE**), and **root mean-squared error** (**RMSE**) in the experiments.

— DKT [34]: This model applies the RNN to the KT problem to capture the hidden knowledge states of each student, and is superior to the BKT. It uses only concepts as inputs.
— DKVMN [48]: It uses key–value memory to track the knowledge state of a student across latent concepts.
— SAKT [31]: It uses an attention mechanism to determine the importance of previously answered exercises in a sequence to predict the upcoming one.
— AKT [11]: It combines the attention model with Rasch model-based embeddings, in which the attention weights exponentially decay w.r.t. the distance between exercises in a sequence to describe a student's forgetting behavior over a long learning period.
— HawkesKT [42]: It integrates a RNN with IRT-based embeddings, where the network's hidden state weights linearly diminish in relation to the time gap between learning activities, to characterize a student's knowledge retention across an extended study duration.
— MAN [17]: It merges memory-augmented neural networks with attention-based neural networks in the MAN, where a context-aware attention mechanism dynamically adjusts the balance between long-term and recent learner knowledge, effectively addressing the "SSP" in e-learning systems.

5.2.2 Training and Testing. We performed standard k-fold cross-validation (with k = 5) on all the models and datasets for evaluation. For each fold, 20% of the response records were used as the test dataset, 20% as the validation set, and 60% as the training set. We truncated sequences of student responses that were longer than 80 to acquire more pertinent data on the local context. If a student's sequence consisted of over 80 exercises, we split it into multiple, shorter sequences. We used the Adam optimizer to train all models with a batch size of 64 students to balance the accuracy of training with the speed of convergence. We implemented all the KT models in PyTorch. An epoch of training took less than 10 seconds. We set the maximum number of epochs to 120.

5.3 Datasets

We used of four publicly accessible datasets to evaluate the predictive accuracy of the ELAKT model against the baseline models. An overview of the datasets is given in Table 2. The first dataset was provided by the ASSISTments online tutoring platform ASSISTments09-10.[1] It contained 325 k responses from 4,151 students when answering exercises involving 110 concepts. The second dataset was ASSISTments2015,[2] which consisted of students' responses in the 2014–2015 school year. It contained 708 k non-repeating records that were generated by the attempts of 19,917 students at answering exercises involving 100 concepts. The third dataset was

[1]https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data
[2]https://sites.google.com/site/assistmentsdata/datasets/2015-assistments-skill-builder-data

Table 3. Comparison of the Experimental Results on four Metrics

Dataset	Metric	ASSISTments09-10	ASSISTments2015	ASSISTments2017	EdNet-KT1-Sub
DKT	AUC	0.7933 ± 0.0080	0.7048 ± 0.0162	0.7034 ± 0.0013	0.6643 ± 0.0062
	ACC	0.7534 ± 0.0086	0.7463 ± 0.0054	0.6868 ± 0.0044	0.6943 ± 0.0055
	MAE	0.2466 ± 0.0086	0.2537 ± 0.0054	0.3132 ± 0.0044	0.3057 ± 0.0055
	RMSE	0.4965 ± 0.0086	0.5037 ± 0.0054	0.5597 ± 0.0039	0.5528 ± 0.0050
SAKT	AUC	0.8039 ± 0.0198	*0.7996 ± 0.0023*	0.7150 ± 0.0021	0.7578 ± 0.0048
	ACC	0.7582 ± 0.0148	0.7727 ± 0.0046	0.6938 ± 0.0033	0.7280 ± 0.0050
	MAE	0.2532 ± 0.0135	0.2273 ± 0.0046	0.3062 ± 0.0033	0.2720 ± 0.0050
	RMSE	0.5031 ± 0.0135	0.4786 ± 0.0049	0.5534 ± 0.0030	0.5215 ± 0.0048
AKT	AUC	0.8221 ± 0.0041	0.7312 ± 0.0017	0.7532 ± 0.0030	0.6657 ± 0.0061
	ACC	0.7749 ± 0.0044	0.7518 ± 0.0014	0.7065 ± 0.0028	0.6956 ± 0.0058
	MAE	0.2217 ± 0.0052	0.2482 ± 0.0014	0.2935 ± 0.0028	0.3044 ± 0.0058
	RMSE	0.4709 ± 0.0055	0.4982 ± 0.0014	0.5418 ± 0.0026	0.5517 ± 0.0052
AKT^o	AUC	*0.8346 ± 0.0036*	0.7828 ± 0.0019	0.7702 ± 0.0026	N/A
DKVMN	AUC	0.8160 ± 0.0050	0.7045 ± 0.0176	0.6863 ± 0.0030	0.6453 ± 0.0053
	ACC	0.7646 ± 0.0059	0.7443 ± 0.0067	0.6808 ± 0.0035	0.6868 ± 0.0058
	MAE	0.8160 ± 0.0050	0.2557 ± 0.0067	0.3192 ± 0.0035	0.3132 ± 0.0058
	RMSE	0.7647 ± 0.0059	0.5056 ± 0.0067	0.5650 ± 0.0031	0.5596 ± 0.0052
HawkesKT	AUC	0.7397 ± 0.0234	N/A[5]	0.7036 ± 0.0009	*0.8869 ± 0.0089*
	ACC	0.7157 ± 0.0106		0.6860 ± 0.0009	0.8337 ± 0.0067
	MAE	0.2843 ± 0.0106		0.3140 ± 0.0009	0.1663 ± 0.0067
	RMSE	0.5331 ± 0.0098		0.5603 ± 0.0008	0.4078 ± 0.0082
MAN[6]	AUC	0.8160 ± 0.0050	0.7669 ± 0.0093	*0.7960 ± 0.0028*	0.8123 ± 0.0131
	ACC	0.7647 ± 0.0059	0.7511 ± 0.0012	0.7350 ± 0.0040	0.8420 ± 0.0063
	MAE	0.2353 ± 0.0059	0.2489 ± 0.0012	0.2650 ± 0.0040	0.2513 ± 0.0063
	RMSE	0.4851 ± 0.0061	0.4989 ± 0.0012	0.5147 ± 0.0039	0.4558 ± 0.0057
ELAKT	AUC	**0.8775 ± 0.0184**	**0.9127 ± 0.0020**	**0.8316 ± 0.0016**	**0.8965 ± 0.0034**
	ACC	0.8063 ± 0.0144	0.8487 ± 0.0014	0.7624 ± 0.0035	0.8210 ± 0.0040
	MAE	0.1937 ± 0.0144	0.1513 ± 0.0014	0.2376 ± 0.0035	0.1790 ± 0.0040
	RMSE	0.4400 ± 0.0167	0.3889 ± 0.0018	0.4875 ± 0.0035	0.4230 ± 0.0047
Improv.	AUC	5.14%	14.14%	4.47%	1.08%

The proposed ELAKT model outperformed all the baseline methods in terms of the AUC, ACC, MAE, and RMSE on all datasets. The best models are shown in **bold**, and the second-best models are given *italics*. The row "Improv". indicates the relative performance improvement of the best model over the second-best model on the AUC metric.

ASSISTments2017.[3] It contained 942,816 responses from 1,709 students and involved 102 concepts. The fourth dataset was EdNet1-Sub.[4] It was a large-scale dataset formulated by the intelligent tutoring system Santa. It consisted of four datasets, named KT1, KT2, KT3, and KT4, that involved different topics. We chose records of interactions of the first 10,000 students in the KT1 dataset from it. The resulting subset was called EdNet1-Sub. These four datasets have been widely used to evaluate the performance of the DKT, SAKT, AKT, and other variants of KT methods.

5.4 Predictive Performance

The experimental results of all methods on all the datasets are presented in Table 3. The AUC, ACC, MAE, and RMSE of all models are presented to evaluate their performance. The ELAKT model

[3] https://sites.google.com/view/assistmentsdatamining/dataset
[4] https://github.com/riiid/ednet
[5] "N/A" is due to the fact that the dataset is not suitable to the method or not prensented in the original paper.
[6] The experimental results obtained under our reproduction of the code and dataset division.

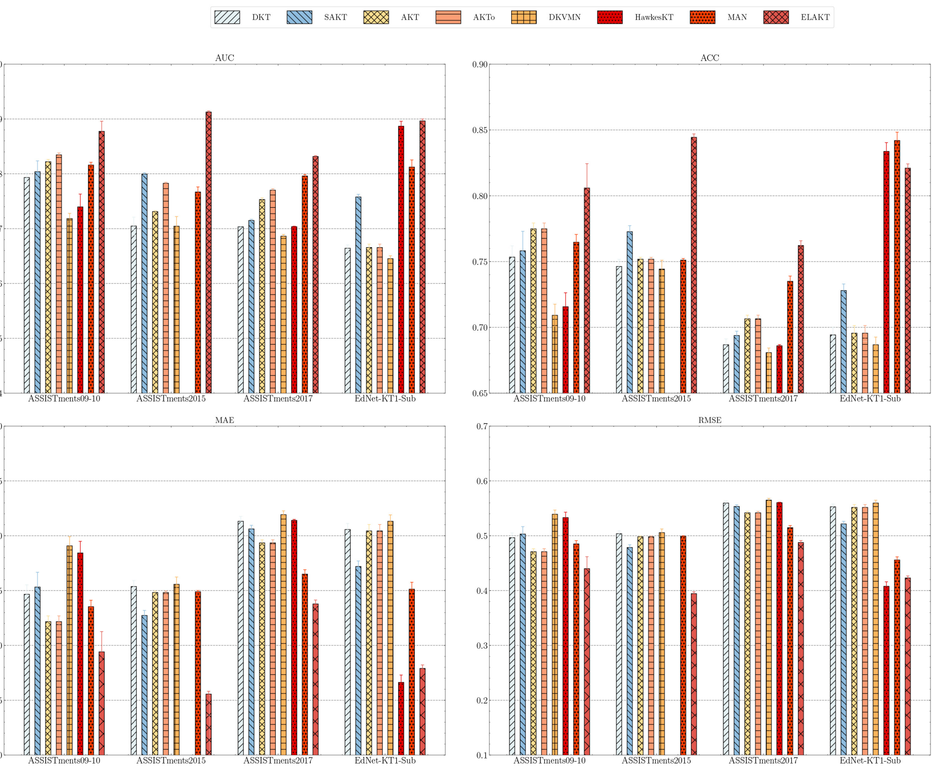


Fig. 4. Comparisons between ELAKT and the baseline techniques in terms of four metrics on four public datasets. The best values of the metrics are marked with asterisks (*).

recorded an AUC of 87.96% on the ASSISTments09-10 dataset, which was higher than those of all the baseline models and constituted an improvement of 5.14% over the best-performing baseline model. It also attained the highest AUC on the ASSISTments2015, ASSISTments2017, and EdNet-KT1-Sub datasets, with improvements of 14.14%, 4.47%, and 1.08% compared with the second-best models, respectively. The row AKT^{o} shows the original results from the relevant study on the AKT [11]. This baseline was added here because according to a report in the official AKT GitHub repository, its performance degraded slightly after a bug had been fixed. The ELAKT model also delivered the best performance on all four datasets in terms of the ACC, MAE, and RMSE. To sum up, the proposed ELAKT model delivered significantly better performance than state-of-the-art DNN-based KT models on all four datasets. We also use histograms to represent the performance of the six models on the four datasets in Figure 4.

5.5 Ablation Study

5.5.1 Experiments on Model Variants. We conducted an ablation study to examine the specific role of each module in the ELAKT framework. To this end, different variants of ELAKT were compared to evaluate the effects of aggregating input knowledge embeddings, aggregating hidden knowledge states, and multi-timestep correction of predictions, respectively, on the performance of the model. These variants are listed as follows:

— **ELAKT-Knowledge Aggregation with Causal Convolution (ELAKT-KACC).** This variant was obtained by replacing the module for aggregating input knowledge embeddings using the encoder with a module for aggregating input knowledge embeddings with causal convolutions. The purpose was to evaluate the impact of choosing *different* methods of aggregating input knowledge on the performance of ELAKT.

- **ELAKT-KACC-NoHidden** (**ELAKT-KACCNH**) Aggregation**.** This variant was created by replacing the encoder module, which aggregates input knowledge embeddings in ELAKT, with a module that uses causal convolutions for aggregation. Additionally, the module for aggregating and smoothing hidden knowledge was removed from ELAKT. The objective was to assess the impact of using different methods for aggregating input knowledge on ELAKT's performance, specifically in the absence of the second causal convolution module.
- **ELAKT-KACC-NoMultiLoss (ELAKT-KACCNML).** This variant was created by replacing the encoder module, which aggregates input knowledge embeddings in ELAKT, with a module that uses causal convolutions for aggregation. Additionally, the variant model did not contain multi-timestep prediction correction. The objective was to assess the impact of using different methods for aggregating input knowledge on ELAKT's performance, specifically in the absence of the multi-timestep prediction correction module.
- **ELAKT-NoAggr (ELAKT-NA).** This variant was obtained by removing the module for aggregating input knowledge embeddings from ELAKT. The purpose was to evaluate the impact of aggregating input knowledge on its performance.
- **ELAKT-NoHidden** (**ELAKT-NH**) Aggregation**.** This variant was obtained by removing the module for aggregating and smoothing hidden knowledge from ELAKT. The purpose was to evaluate the impact of the second causal convolution module on its performance.
- **ELAKT-NoMultiLoss (ELAKT-NML).** This variant did not contain multi-timestep prediction correction, and was used to verify the impact of leveraging the loss in predictive accuracy over multiple timesteps on the performance of ELAKT.
- **ELAKT-NoAggr-NoHiddenAggregation (ELAKT-NANH).** This variant was obtained by removing both the module to aggregate input knowledge embeddings and that to aggregate and smooth hidden knowledge states from ELAKT. The aim was to evaluate their impact on its performance.
- **ELAKT-NoAggr-NoMultiLoss (ELAKT-NANML).** This variant was obtained by removing both the module to aggregate input knowledge embeddings and that to correct predictions over multiple timesteps from ELAKT. It was used to evaluate their joint impact on performance.
- **ELAKT-NoHiddenAggregation-NoMultiLoss (ELAKT-NHNML).** This variant was obtained by removing both the module to aggregate and smooth hidden knowledge states and that to correct predictions over multiple timesteps from ELAKT. It was used to evaluate their joint impact on performance.
- **ELAKT-NoHiddenAggregation-NoMultiLoss-Knowledge Aggregation With Causal Convolution (ELAKT-NHNML-KACC).** This variant was obtained by removing both the module to aggregate and smooth hidden knowledge states and that to correct predictions over multiple timesteps from ELAKT, and by replacing the aggregation of input knowledge embeddings through an encoder with that based on causal convolutions. This variant is equivalent to adding knowledge aggregation with causal convolutions to the traditional SAKT, which is a considerably downgraded version of ELAKT.
- **ELAKT-NoAggr-NoHiddenAggregation-NoMultiLoss (ELAKT-NANHNML).** This variant was obtained by removing all modules from ELAKT, which caused it to reduce to SAKT. This variant was used as a baseline model for comparison with the other variants.

The models of KACC, KAENC, ASHKS, and MPC in Table 4 were "input aggregations of knowledge embeddings with causal convolutions", "input aggregated knowledge embeddings by using

Table 4. Results of the Ablation Study

No.	Model	Component				Dataset			
		KACC	KAENC	ASHKS	MPC	ASSISTments09-10	ASSISTments2015	ASSISTments2017	EdNet-KT1-Sub
1	ELAKT	×	✓	✓	✓	0.8629	0.9140	0.8340	0.8931
2	ELAKT-KACC	✓	×	✓	✓	0.8055	0.8303	0.7559	0.7880
3	ELAKT-KACCNH	✓	×	✓	×	0.8091	0.8234	0.7398	0.7766
4	ELAKT-KACCNML	✓	×	×	✓	0.8089	0.8199	0.7511	0.7857
5	ELAKT-NA	×	×	✓	✓	0.8098	0.8476	0.7575	0.7831
6	ELAKT-NML	×	✓	✓	×	**0.8575**	**0.9056**	**0.8273**	**0.8852**
7	ELAKT-NANH	×	×	×	✓	0.8061	0.8172	0.7383	0.7678
8	ELAKT-NANML	×	×	✓	×	0.8061	0.8481	0.7447	0.7848
9	ELAKT-NHNML	×	✓	×	×	*0.8364*	*0.8957*	*0.7952*	*0.8825*
10	ELAKT-NHNMLKACC	✓	×	×	×	0.8005	0.8090	0.7304	0.7671
11	ELAKT-NANSNML	×	×	×	×	0.7998	0.8060	0.7292	0.7633

The best variants are presented in **bold**, and the second-best variants are shown in *italics*.

an encoder", "aggregating and smoothing hidden knowledge states", and "multi-timestep prediction correction", respectively. The four datasets used above were applied again, and the results are presented in Table 2. Owing to the large scale of the datasets, we selected only the first fold of the five-fold data for the ablation experiments. The values of $kernel\_size$ of the casual convolution in "aggregating and smoothing hidden knowledge states" and "aggregating input knowledge embeddings" were both set to five by default. The former is denoted by n and the latter by e (see Table 1 for recall). We explored the impact of choosing different values of n and e on the predictive performance of ELAKT and its variants.

The experimental results show that the "KAENC" module had the greatest impact on the predictive performance of the model. The model equipped with this module significantly outperformed the counterpart model without "KAENC", especially on the "EdNet-KT1-Sub" dataset (see No. 9 vs. No. 11). Compared with the baseline model ELAKT-NANSNML (No. 11), the ASHKS, KACC, and MPC modules each improved the predictive performance of the model, but the gains due to them were limited (see No. 8, No. 10, and No. 7 compared with No. 11).

When these four modules were combined in pairs, the combination of the "KAENC" module and the "ASHKS" module yielded the best predictive performance among all variants on all four datasets (see No. 6 vs. No. 11). The "KACC" module and the "MPC" module also significantly improved the predictions (see No. 2 and No. 7 compared with No. 11). However, the combination of the "ASHKS" and "MPC" modules led to only moderate improvements in performance (see No. 5 vs. No. 11). The "KACC" module as well as variants generated by combining the "KACC" with other modules improved predictions compared with the baseline model (see No. 2 vs. No. 11). However, the combination of KACC with other modules did not present a significant improvement in predictive performance, and in some cases, even led to a reduction in efficacy (see No. 2 vs. No. 3 and No. 4). The KAENC module led to much greater improvement than the KACC module (see No. 1 vs. No. 11). These results can be explained in the following way: When the proposed "KAENC" module was not used, the knowledge state output at each timestep represented only the student's performance on the current exercise rather than their comprehensive knowledge state. The convolution of knowledge states between exercises could not be smoothed because these states might not have been related to one another.

5.5.2 Hyper-Parameter Tuning. In Sections 4.2 and 4.3, we defined the kernel size e for the causal convolution of the module to aggregate and smooth hidden knowledge states, and the kernel size n for the causal convolution of the module to input aggregated knowledge embeddings with causal convolutions. These two hyper-parameters should be fine-tuned during the implementation of the model. A larger e represents a wider range for the aggregation and smoothing of the knowledge state. We set e to one of $\{1, 2, 3, 4, 5, 6\}$. Similarly, a larger n represents a wider range for the aggregation of contextual knowledge embeddings. We set n to one of $\{0, 1, 2, 3, 4, 5, 6\}$ to conduct a grid search for the "optimal" hyper-parameters across models. When $n = 0$, this implies

Table 5. Average AUC Scores on the four Public Datasets, Obtained by using Different Combinations of e and n

Datasets	kernel size (n, e)	$e = 1$	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$
ASSISTments 09-10	$n = 0$	0.8480	0.8629	0.8667	0.8664	0.8629	**0.8732**
	$n = 1$	0.8061	0.8144	*0.8155*	0.8096	0.8098	0.8134
	$n = 2$	0.8027	0.8097	0.8063	0.8116	0.8130	0.8095
	$n = 3$	0.8032	0.8104	0.8026	0.8056	0.8101	0.8100
	$n = 4$	0.7999	0.8060	0.8040	0.8018	0.8037	0.8062
	$n = 5$	0.7992	0.8095	0.8007	0.8044	0.8055	0.8110
	$n = 6$	0.8005	0.8048	0.8004	0.8084	0.8072	0.8032
ASSISTments 2015	$n = 0$	0.8989	0.9118	0.9102	0.9100	**0.9140**	0.9107
	$n = 1$	0.8172	0.8435	0.8294	0.8460	*0.8476*	0.8459
	$n = 2$	0.8149	0.8342	0.8378	0.8415	0.8434	0.8430
	$n = 3$	0.8163	0.8351	0.8309	0.8399	0.8435	0.8426
	$n = 4$	0.8150	0.8268	0.8324	0.8334	0.8372	0.8356
	$n = 5$	0.8238	0.8317	0.8274	0.8341	0.8303	0.8299
	$n = 6$	0.8147	0.8231	0.8323	0.8301	0.8339	0.8348
ASSISTments 2017	$n = 0$	0.8076	0.8227	0.8306	0.8328	0.8338	**0.8356**
	$n = 1$	0.7383	0.7503	0.7547	0.7496	0.7575	0.7548
	$n = 2$	0.7440	0.7537	0.7538	*0.7646*	0.7420	0.7496
	$n = 3$	0.7464	0.7487	0.7490	0.7590	0.7560	0.7543
	$n = 4$	0.7469	0.7592	0.7561	0.7565	0.7601	0.7564
	$n = 5$	0.7373	0.7547	0.7632	0.7503	0.7559	0.7579
	$n = 6$	0.7385	0.7498	0.7580	0.7596	0.7600	0.7575
EdNet1-Sub	$n = 0$	0.8814	0.8889	0.8901	0.8929	0.8931	**0.8952**
	$n = 1$	0.7678	0.7773	0.7897	0.7804	0.7831	0.7817
	$n = 2$	0.7753	0.7790	0.7872	0.7811	0.7861	0.7832
	$n = 3$	0.7722	0.7806	0.7821	0.7854	0.7870	0.7868
	$n = 4$	0.7744	0.7877	0.7901	0.7842	0.7861	*0.7956*
	$n = 5$	0.7751	0.7842	0.7875	0.7900	0.7880	0.7893
	$n = 6$	0.7760	0.7831	0.7849	0.7883	0.7897	0.7909

The combinations of hyper-parameters (n, e) with the best AUC with are given in **bold**, and those with the second-best are given in *italics*.

that the ELAKT model utilizes the "KAENC" module for aggregating input knowledge embeddings, while $n \neq 0$ means that the model uses the "KACC" module to this end. The "multi-timestep prediction correction" module was used in all experiments to tune the hyper-parameters. Table 5 shows the experimental results in terms of the AUC scores, which show that increasing the value of e enhanced the predictive performance of the ELAKT model.

When $n = 0$, increasing the hyper-parameter e led to a significant improvement in the predictive performance of the model on the ASSISTments09-10 dataset. The increase in n led to almost no improvement in its performance on ASSISTments09-10. Moreover, increasing e when $n \neq 0$ led to only minor enhancements in performance.

The results on the ASSISTments 2015 dataset indicate that increasing the value of e enhanced the model's predictive performance. However, while small increases in n yielded a minor improvement in performance, further increases tended to reduce the model's predictive capacity. Excessively large values of n even negatively impacted the effectiveness of e in boosting performance. In most experiments on e, the model achieved its optimal predictive performance when e was set to five.

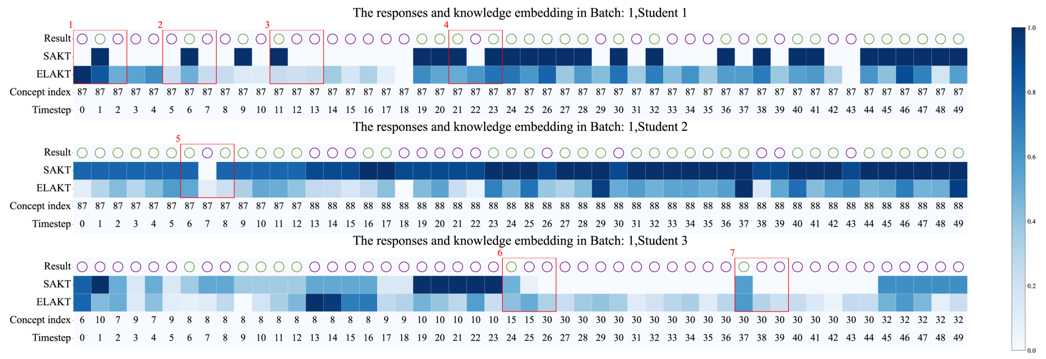


Fig. 5. Visualizing the knowledge embeddings derived from the response sequences of three students through SAKT and ELAKT from ASSISTments09-10. The aggregated knowledge embeddings employed by the ELAKT remained stable relative to the extreme changes in the knowledge embeddings used by SAKT.

The experimental results on both the ASSISTments 2017 dataset and the EdNet-KT1-Sub dataset showed similar trends to the above, whereby increases in the hyper-parameters e and n improved the predictive performance of the model. When $\{n = 2, e = 4\}$ and $\{n = 4, e = 6\}$, the variants using the "KACC" module achieve optimal predictive performance on the ASSISTments 2017 and the EdNet-KT1-Sub datasets, respectively.

The results of the experiments on tuning the hyper-parameters can be summarized as follows:

— Reasonable values of the hyper-parameter e significantly enhanced the model's predictive performance while excessively large values reduced it. This can be attributed to the fact that over-smoothing the knowledge state introduces irrelevant concept states, potentially leading to the use of incorrect concept states when predicting the student's response.

— Although both the "KAENC" and "KACC" modules can be used to aggregate input knowledge embeddings in the ELAKT model, the predictive performance of the model is significantly better when using the "KAENC" module. This can be attributed to its higher flexibility of aggregation and the accuracy of the attention mechanism. The causal convolution is incapable of this because it requires a convolution of a fixed size.

6 VISUALIZATION

6.1 Visualizing Aggregation of Input Knowledge Embeddings

To evaluate the effectiveness of the process of aggregating the input knowledge embeddings, we visualized vectors of the knowledge embeddings of both SAKT and ELAKT. The steps are as follows:

(1) Calculate the L2-norm of each row of the knowledge embedding $\boldsymbol{K}^{\text{aggr}} \in \mathbb{R}^{T*D_m}$ (T is the length of the sequence, defined in Algorithm 1, and D_m is the number of dimensions of embeddings of the model) to generate a vector in $\mathbb{R}^T$ for both SAKT and ELAKT;

(2) Normalize the elements in the vector to a range between zero and one;

(3) Visualize the normalized vector by using a heatmap.

The first 50 elements (i.e., timesteps or exercise items) in the normalized vectors as well as the index of concepts and the answers (results) for both SAKT and ELAKT are presented in Figure 5.

The top row of each heatmap in Figure 5 displays the sequence of the student's answers: The light-green color represents a correct answer while purple represents an incorrect one. The second and third rows present the L2-norm of the knowledge embeddings in the SAKT model and the aggregated knowledge embeddings in the ELAKT models, respectively, with the intensity of the colors indicating the value of the normalized L2-norm. The fourth row shows the index of concepts associated with the given exercise at the given timestep (in the fifth row). In all experiments, the length of the sequence T of the model was set to 80, with zero padding applied to incomplete sequences. Due to the difficulty of visualizing long sequences, only the first 50 exercise items are displayed here.

Figure 5 shows that the SAKT model underwent fluctuations in the L2-norm of knowledge embeddings. It jumped from the maximum to the minimum value, simply because of the occasional correct or incorrect answer (see red box, Nos. 1–5). When a student consistently answered exercises incorrectly, the overall L2-norm of the knowledge embedding even approached zero (see red box, Nos. 6–7). By contrast, the L2-norm of the aggregated knowledge embeddings generated by the ELAKT model was smooth, with few extreme changes. Even if a student answered multiple exercises incorrectly, the aggregated knowledge embedding remained stable. This suggests that module to aggregate input knowledge embeddings was effective in consolidating contextual knowledge embeddings and obtaining comprehensive ones. This feature is consistent with the fact that a student's knowledge state should not change drastically over a short time, even though abnormal behaviors may occasionally occur.

6.2 Visualizing Attention Weights

To evaluate the impact of aggregated knowledge embeddings on the encoder module detailed in Section 4.3, we randomly selected a student, and visualized its attention weights in $head_2$ and $head_4$ (the default value of the number of attention heads was four) in both the SAKT and the ELAKT models based on data from the ASSISTments09-10 dataset.

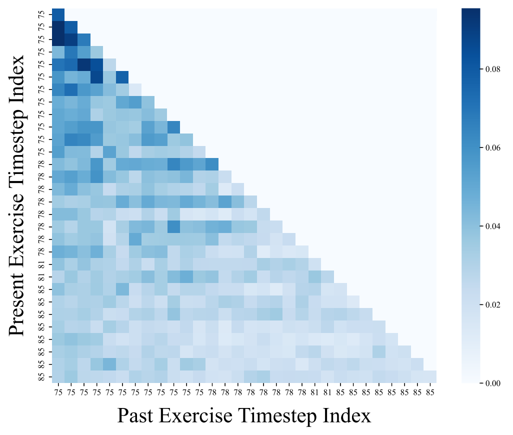
Figures 6(a) and 6(b) present the results of visualization of attention obtained by the ELAKT model, while Figures 6(c) and 6(d) show those of the SAKT model. Figure 6 shows that the attention weights of the SAKT model significantly fluctuated. While a few points had high attention values, most timesteps had low attention values. By contrast, the attention weights of the ELAKT model were smoother. Its pattern of attention weights was a more appropriate representation of the student's comprehensive knowledge state, and it could make local adjustments based on the actual knowledge embedding at each timestep.

6.3 Visualizing the State of the Full Knowledge Concept for Prediction

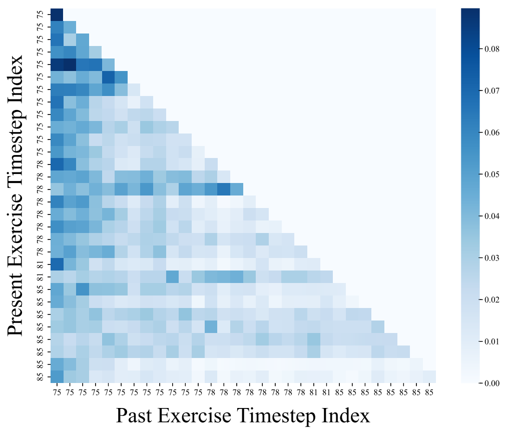
To verify the capability of ELAKT to trace the states of the KCs, we present a visualization in Figure 7. In Section 4.3, we use a mapping of KCs to transform the student's hidden knowledge state $\boldsymbol{c_t}$ into the comprehensive state of KCs $\boldsymbol{s_t}$. And then, the $p_t \in [0, 1]^L$, which was obtained by passing $\boldsymbol{s_t}$ through the sigmoid activation function, can be interpreted as the response probability at timestep t for all KCs. The figure illustrates the evolution of p_t for a student in the first batch of the ASSISTments09-10 dataset over a course of 20 timesteps. During these 20 timesteps, five KCs were used, with indices of 8 (pentagon), 9 (rhombus), 10 (triangle), 55 (square), and 87 (circle).

The first two rows in Figure 7 represent the student's actual responses (i.e., the ground-truth response) at each timestep, and the predicted responses made by the model based on the current state of the student's KCs (see green box), respectively. In the first two rows, the shape with a rectangular border means an incorrect response and the shape without a border means a correct response. When both the shapes at the same time step are with a boarder or without a boarder, concurrently, this indicates a successful prediction. Difference in terms of the boarder indicates a failure in prediction. The shapes represent different KCs.

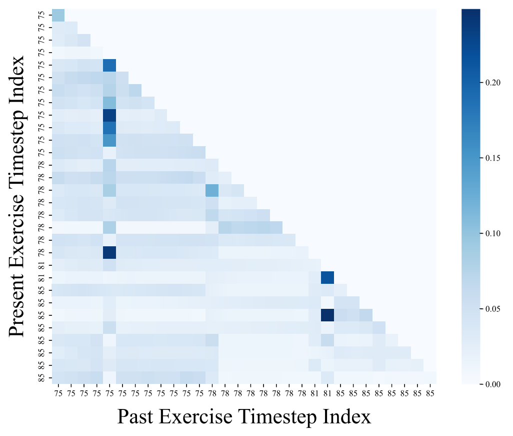
The five rows within the blue box illustrate the evolution of the response probabilities with respect to the five KCs over time. The depth of the grayscale blocks was determined by the



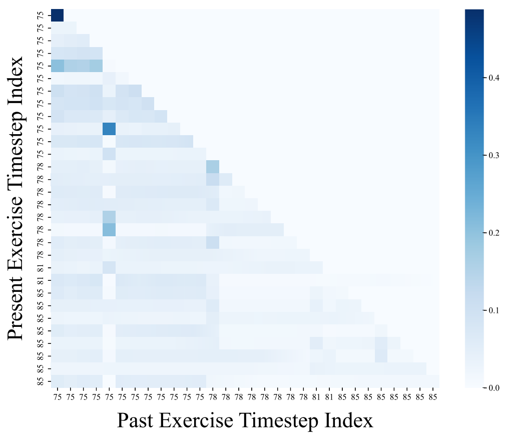
(a) Self-attention-based visualization of ELAKT in $head_2$



(b) Self-attention-based visualization of ELAKT in $head_4$



(c) Self-attention-based visualization of ELAKT in $head_2$



(d) Self-attention-based visualization of SAKT in $head_4$

Fig. 6. The self-attention-based visualizations of both SAKT and ELAKT in the module to aggregate and smooth hidden knowledge states on the ASSISTments 09-10 dataset.

normalized 0–1 value (darker blocks represent higher probabilities of the student correctly answering the exercises involving a certain KC). Again, different shapes embedded in the block represent different KCs, and their colors indicate the predicted outcomes for an exercise involving the respective KC. If the value of the state of the KC exceeded 0.5, the embedded shape was colored,indicating that the model will predict the student to answer the exercise correctly; if the value was below 0.5, the embedded shape was black, and reflected an incorrect response.

We define the failure of the prediction as a discrepancy between the predicted response based on the knowledge state and the actual response (ground truth). The prediction for "concept 8" was correct once and failed once in the first two timesteps. When the response was incorrect in the first timestep, the corresponding state of the KC decreased (block with a light grey; see red box No. 1). When the follow-up response to the exercise involving "concept 8" was correct (in time step 2), the state of the corresponding KC increased in the third timestep. From the third to the seventh timesteps, the predictions for exercises involving "concept 9" were consistently successful (The colors of "Prediction result" and "Ground truth" match perfectly from the third to

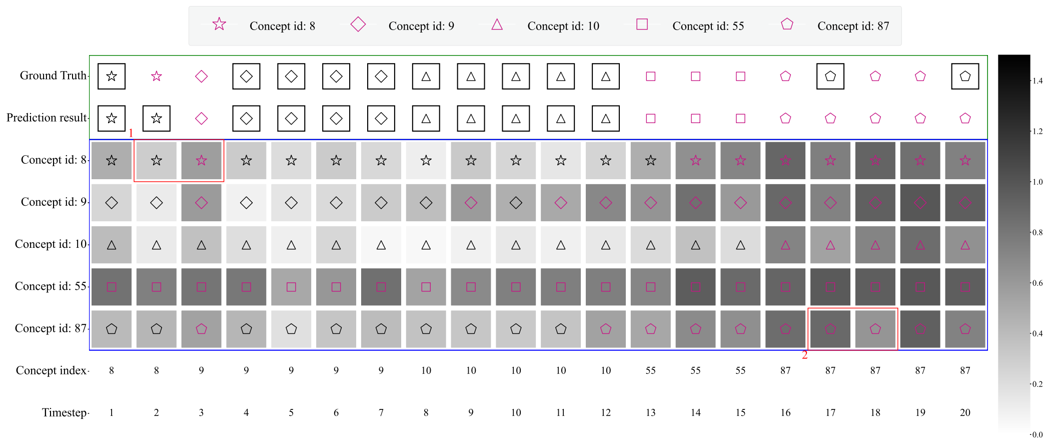


Fig. 7. Visualizing the tracing of the state of the KCs of a student in ASSISTments09-10. In the first two rows, the shape with a rectangular border means an incorrect response and the shape without a border means a correct response. Three failure in predictions occur at the timestep 3, 17, and 20.

the seventh timesteps). Predictions for exercises, involving concepts 10 and 55, respectively, were also successful. Two failed predictions were obtained for "concept 87". Nonetheless, the state of the corresponding KC decreased (see red box No. 2) when the student's response was incorrect. It is demonstrated that one or two incorrect responses did not have a significant impact on the comprehensive state of the KC used by the ELAKT. There were three failed predictions out of the total 20 predictions, giving an accuracy of 85%.

7 CONCLUSIONS AND FUTURE WORK

In this article, we addressed two main limitations in KT methods: the difficulty of capturing a comprehensive knowledge state at each timestep, and the lack of consideration of stochastic behaviors, such as students' slipping and guessing behaviors. To address these issues, we developed a KT model, called ELAKT, based on the framework of the self-attentive transformer. The proposed model can trace the knowledge states of students at each timestep while predicting their performance on the upcoming exercises. It uses an encoder module to aggregate knowledge embeddings, and further aggregates and smooths hidden knowledge states by using causal convolutions. The results of experiments on four real-world datasets demonstrated that the ELAKT model achieves state-of-the-art predictive performance, while those of ablation studies validated the impact of each of the three proposed modules on the overall performance. The combination of the comprehensive tracing of KCs and the predictive power of the ELAKT model can help design better intelligent tutoring applications. Acquiring a comprehensive knowledge state enables more extensive and holistic prediction of student responses in future assessments, because it involves the consideration of historical exercise-response pairs of a student and the complete set of KCs. This comprehensive quantity can help significantly enhance personalized learning resource recommendations. In future work, we plan to enhance the textual interpretability of the ELAKT model by using large language models, such as ChatGPT, for pre-processing the dataset to obtain more accurate KC–exercise relationships as well as representative text labels for the KCs.

ACKNOWLEDGMENTS

The authors are affiliated with both Beihang University and Zhongguancun Laboratory, which contributed equally to this work.

REFERENCES

[1] Ghodai Abdelrahman and Qing Wang. 2019. Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 175–184.

[2] Ghodai Abdelrahman and Qing Wang. 2022. Deep graph memory networks for forgetting-robust knowledge tracing. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7844–7855.

[3] Ghodai Abdelrahman, Qing Wang, and Bernardo Pereira Nunes. 2022. Knowledge tracing: A survey. *ACM Computing Surveys* 55, 11, (2022), 1–37.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. arXiv:1607.06450 . Retrieved from https://arxiv.org/abs/1607.06450

[5] Alan D. Baddeley and Graham Hitch. 1993. The recency effect: Implicit learning with explicit retrieval? *Memory & Cognition* 21, 2 (1993), 146–155.

[6] Jiayi Chen, Wen Wu, and Liang He. 2022. C3SASR: Cheap causal convolutions for self-attentive sequential recommendation. arXiv:2211.01297 . Retrieved from https://arxiv.org/abs/2211.01297

[7] Penghe Chen, Yu Lu, Vincent W. Zheng, and Yang Pian. 2018. Prerequisite-driven deep knowledge tracing. In *Proceedings of the 2018 IEEE International Conference on Data Mining*. IEEE, 39–48.

[8] Youngduck Choi, Youngnam Lee, Junghyun Cho, Jineon Baek, Byungsoo Kim, Yeongmin Cha, Dongmin Shin, Chan Bae, and Jaewe Heo. 2020. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the 7th ACM Conference on Learning@ Scale*. 341–344.

[9] Albert T. Corbett and John R. Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction* 4, 4 (1994), 253–278.

[10] Jiajun Cui, Zeyuan Chen, Aimin Zhou, Jianyong Wang, and Wei Zhang. 2023. Fine-grained interaction modeling with multi-relational transformer for knowledge tracing. *ACM Transactions on Information Systems* 41, 4 (2023), 1–26.

[11] 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*. 2330–2339.

[12] José González-Brenes, Yun Huang, and Peter Brusilovsky. 2014. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *Proceedings of the 7th International Conference on Educational Data Mining*. University of Pittsburgh, 84–91.

[13] José P. González-Brenes and Jack Mostow. 2012. Dynamic cognitive tracing: Towards unified discovery of student and cognitive models. *International Educational Data Mining Society* (2012).

[14] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. arXiv:1410.5401 . Retrieved from https://arxiv.org/abs/1410.5401

[15] Rebeen Ali Hamad, Masashi Kimura, Longzhi Yang, Wai Lok Woo, and Bo Wei. 2021. Dilated causal convolution with multi-head self attention for sensor human activity recognition. *Neural Computing and Applications* 33, 20 (2021), 13705–13722.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[17] Liangliang He, Xiao Li, Pancheng Wang, Jintao Tang, and Ting Wang. 2023. MAN: Memory-augmented attentive networks for deep learning-based knowledge tracing. *ACM Transactions on Information Systems* 42, 1 (2023), 1–22.

[18] Liangliang He, Jintao Tang, Xiao Li, Pancheng Wang, Feng Chen, and Ting Wang. 2022. Multi-type factors representation learning for deep learning-based knowledge tracing. *World Wide Web* 25, 3 (2022), 1343–1372.

[19] Zhankui He, Handong Zhao, Zhe Lin, Zhaowen Wang, Ajinkya Kale, and Julian McAuley. 2021. Locker: Locally constrained self-attentive sequential recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3088–3092.

[20] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[21] Mohammad Khajah, Robert V. Lindsey, and Michael C. Mozer. 2016. How deep is knowledge tracing? arXiv:1604.02416 . Retrieved from https://arxiv.org/abs/1604.02416

[22] M. Khajah, R. V. Lindsey, and M. C. Mozer. 2016. How deep is knowledge tracing? *Proceedings of EDM* (2016), 94–101.

[23] Kenneth R. Koedinger, John C. Stamper, Elizabeth A. McLaughlin, and Tristan Nixon. 2013. Using data-driven discovery of better student models to improve student learning. In *Proceedings of the International Conference on Artificial Intelligence in Education*. Springer, 421–430.

[24] Dan Kondratyuk, Liangzhe Yuan, Yandong Li, Li Zhang, Mingxing Tan, Matthew Brown, and Boqing Gong. 2021. Movinets: Mobile video networks for efficient video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16020–16030.

[25] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Proceedings of Advances in Neural Information Processing Systems* (2019), 1598–1607.

[26] Yanan Li, Haitao Yuan, Zhe Fu, Xiao Ma, Mengwei Xu, and Shangguang Wang. 2023. ELASTIC: Edge workload forecasting based on collaborative cloud-edge deep learning. In *Proceedings of the ACM Web Conference 2023*. ACM, 3056–3066.

[27] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open* (2022).

[28] Congjie Liu and Xiaoguang Li. 2021. Multi-factor memory attentive model for knowledge tracing. In *Proceedings of the Asian Conference on Machine Learning*. PMLR, 856–869.

[29] Larry R. Medsker and LC Jain. 2001. Recurrent neural networks. *Design and Applications* 5 (2001), 64–67.

[30] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. arXiv:1609.03499 . Retrieved from https://arxiv.org/abs/1609.03499

[31] Shalini Pandey and George Karypis. 2019. A self-attentive model for knowledge tracing. arXiv:1907.06837 . Retrieved from https://arxiv.org/abs/1907.06837

[32] Shalini Pandey and Jaideep Srivastava. 2020. RKT: Relation-aware self-attention for knowledge tracing. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1205–1214.

[33] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. arXiv:1606.01933 . Retrieved from https://arxiv.org/abs/1606.01993

[34] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Proceedings of the Advances in Neural Information Processing Systems*. 505–513.

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

[36] John C. Stamper and Kenneth R. Koedinger. 2011. Human-machine student model discovery and improvement using datashop. In *Proceedings of the Artificial Intelligence in Education - 15th International Conference, AIED 2011, Auckland, New Zealand, June 28 - July 2011*.

[37] Jianwen Sun, Rui Zou, Ruxia Liang, Lu Gao, Sannyuya Liu, Qing Li, Kai Zhang, and Lulu Jiang. 2022. Ensemble knowledge tracing: Modeling interactions in learning process. *Expert Systems with Applications* 32 (2022), 1–12.

[38] Kikumi K. Tatsuoka. 1983. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement* 20, 4 (1983), 345–354.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*. 5998–6008.

[40] George C. Velmahos, Konstantinos G. Toutouzas, Lelan F. Sillin, Linda Chan, Richard E. Clark, Demetrios Theodorou, and Fredric Maupin. 2004. Cognitive task analysis for teaching technical skills in an inanimate surgical skills laboratory. *American Journal of Surgery* 187, 1 (2004), 114–119.

[41] Bandhav Veluri, Justin Chan, Malek Itani, Tuochao Chen, Takuya Yoshioka, and Shyamnath Gollakota. 2022. Real-time target sound extraction. arXiv:2211.02250 . Retrieved from https://arxiv.org/abs/2211.02250

[42] 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*. 517–525.

[43] Chun-Kit Yeung and Dit-Yan Yeung. 2018. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the 5th Annual ACM Conference on Learning at Scale*. 1–10.

[44] Haitao Yuan and Guoliang Li. 2021. A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering* 6, 1 (2021), 63–85.

[45] Haitao Yuan, Guoliang Li, and Zhifeng Bao. 2022. Route travel time estimation on a road network revisited: Heterogeneity, proximity, periodicity and dynamicity. *Proceedings of the VLDB Endowment* 16, 3 (2022), 393–405.

[46] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2020. Effective travel time estimation: When historical trajectories over road networks matter. In *Proceedings of the 2020 International Conference on Management of Data*. ACM, 2135–2149.

[47] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An effective joint prediction model for travel demands and traffic flows. In *Proceedings of the 37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 348–359.

[48] 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*. 765–774.

[49] Yan Zhao, DeLiang Wang, Buye Xu, and Tao Zhang. 2020. Monaural speech dereverberation using temporal convolutional networks with self attention. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 28 (2020), 1598–1607.

Received 7 June 2023; revised 3 January 2024; accepted 6 March 2024