Self-KT: Self-attentive Knowledge Tracing with Feature Fusion Pre-training in Online Education

Guoqiang Lu[1], Ke Niu[1,*], Xueping Peng[2], Yuhang Zhou[1], Ke Zhang[1], Wenjuan Tai[1]
[1]Computer School, Beijing Information Science and Technology University, Beijing, China
[2]University of Technology Sydney, Australia
Emails:{luguoqiang, niuke}@bistu.edu.cn, xueping.peng@uts.edu.au,
{zhouyuhang, zhangke, taiwenjuan}@bistu.edu.cn

***Abstract*—The goal of the Knowledge Tracing (KT) task is to accurately predict a student's aptitude in answering the next question based on their previous responses. Recent studies have shown promising results by employing pre-training models to capture general feature representations between questions and skills, and subsequently fine-tuning these models for the KT task. However, these methods still face challenges in accurately representing question difficulty and fail to consider the impact of feature fusion during pre-training. Additionally, existing models do not effectively harness the high-level semantic information available after pre-training during fine-tuning, resulting in an underutilization of their potential applications. To this end, this paper proposes Self-attentive Knowledge Tracing (Self-KT) with Feature Fusion Pre-training in the Online Education domain to address these challenges. Self-KT introduces a novel representation of question difficulty and innovatively implements dynamic feature fusion to obtain question embeddings. Furthermore, it enhances the self-attention mechanism by considering the influence of subsequent questions on the current question. We implemented Self-KT on multiple publicly available datasets, and the results demonstrated its significant superiority over the current state-of-the-art methods in knowledge tracing.**

***Index Terms*—Knowledge tracing, Deep neural network, Pre-training, Self-attention, Online education**

I. Introduction

Knowledge tracing (KT) [1] has gained significance in the field of online education. This approach investigates the essential characteristics of questions and skills, analyzes students' learning history and patterns of question-answering, and predicts their probability of correctly answering future questions. Effectively capturing the relationship between questions and skills has emerged as a major challenge in knowledge-tracking tasks, as exercises and skills play a foundational role in this domain [2]–[4].

Fig. 1 visually represents three key challenges that arise during the design of knowledge-tracking models. Researchers have proposed various solutions, particularly deep KT models, which have shown improved effectiveness. These models employ different approaches for feature extraction related to questions and skills. One approach involves assessing the student's skill mastery to model their ability [5]. Another approach focuses on leveraging question-related information to learn the student's ability [6]. Although both methods have demonstrated promising results, accurately predicting a student's knowledge status for specific questions is crucial due to the varying difficulty levels of questions associated with the same skill. To address these challenges, PEBG [7] introduced a pre-training model that utilizes question embeddings to extract features. This approach and subsequent pre-training models have significantly improved upon prior deep KT models by extracting shared feature representations for questions and skills. However, these pre-training methods still lack accuracy in characterizing question difficulty and incorporating feature fusion. In Case I of Fig. 1, PEBG only utilizes the average initial response time and answer type to depict question difficulty. MF-DAKT [8] suggests enhancing the representation of question difficulty by incorporating accuracy rate but relying solely on this information fails to accurately capture question difficulty. As illustrated in Case II of Fig. 1, the features extracted in KT tasks are diverse. Traditional methods often overlook the interaction between features when combining them, resulting in an incomplete capture of significant disparities. In the actual process of student answering, subsequent questions often have a profound impact on the current question, and the current question may inspire modifications to previous questions. As shown in Case III in Fig. 1, during the fine-tuning of the KT task, existing models fail to effectively link the performance of past tasks, which leads to poor predictive results.

In order to address the limitations of existing models, this paper firstly proposes a new method called **Self**-attentive **K**nowledge **T**racing (Self-KT) with feature fusion pre-training guided by the attention factorization machine. The approach distinguishes between subjective and objective difficulties. Subjective difficulty is determined by analyzing the student's response time and number of attempts, while objective difficulty is evaluated based on error rate and question type. By taking into account both subjective cognition and objective features, our method improves the characterization of question difficulty.

Secondly, to assign weights effectively to different features and capture their varying importance, the Attention Factorization Machine (AFM) [9] is employed for feature fusion. AFM calculates attention scores to allocate weights to features, preserves question difficulty information, and learns

*Corresponding author

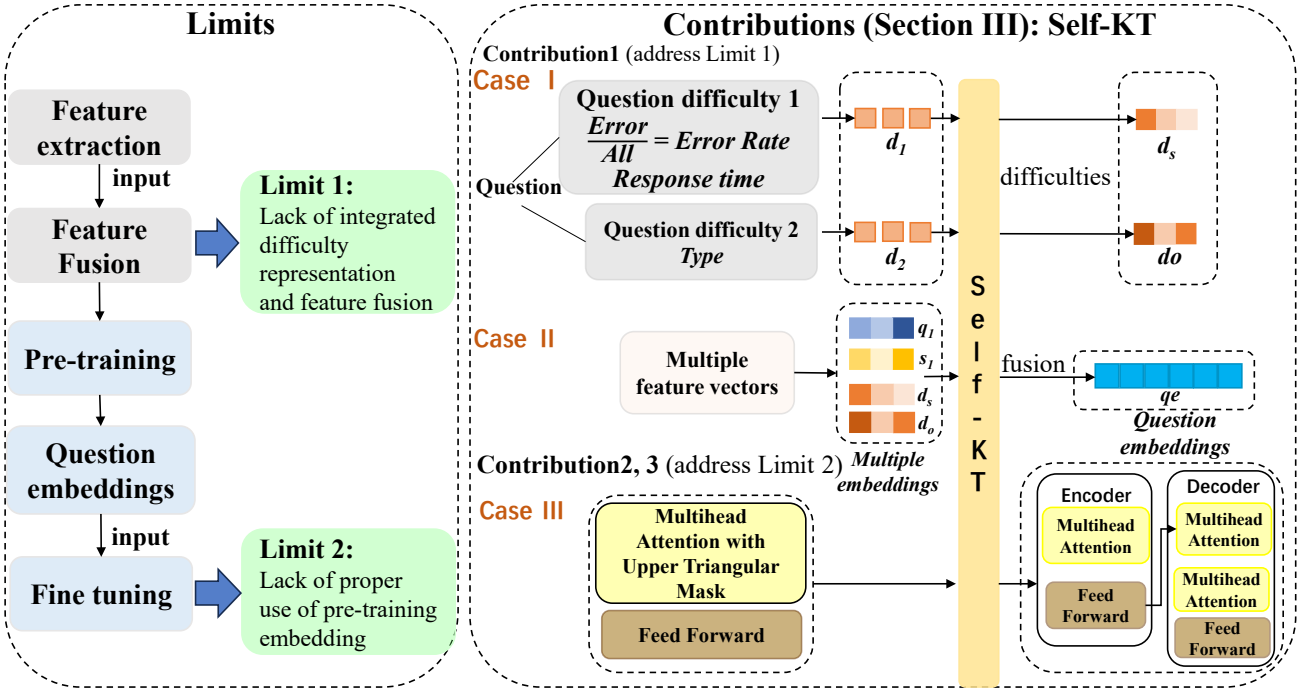


Fig. 1: Our model addresses several limitations of pre-trained and fine-tuned models, and makes the following contributions: 1. We address the lack of question difficulty representation and feature fusion in both Case I and Case II. 2. We address the absence of pre-trained question embedding models that can be applied effectively in Case III.

the relationship between questions and skills. This results in the creation of embedded representations of questions.

Thirdly, in order to preserve the relationship between questions and skills during the prediction process, we utilize an autoencoder that consists of a self-attention mechanism without masking. By eliminating the masking, the model is able to attend to information from subsequent questions, taking into account factors that may trigger modifications to the current predicted question. Furthermore, we introduce lateral connections between the encoder and decoder, connecting the output of each self-attention layer in the encoder to the corresponding input of the self-attention layer in the decoder. This facilitates the transmission and fusion of information across layers, thereby further enhancing the performance of the model [10].

Lastly, we conducted experiments on various real-world datasets, and the results demonstrate that Self-KT outperforms existing KT models. Additionally, several ablation studies were carried out to confirm the role of the key components of Self-KT.

The contributions of our paper can be summarized as follows:

- To address the issue of inaccurate question difficulty features and the allocation of weights in feature fusion, we propose a unique method for characterizing difficulty. Additionally, we enhance the feature fusion module to improve the rationality of weight allocation among different features.
- We propose a novel method for knowledge-tracking prediction that utilizes a self-encoder comprising multi-head attention and feed-forward networks. This approach enables the transmission and fusion of information across layers, maximizing the retention of valuable information in the learned sequence after pre-training.
- To maximize the restoration of the student's answering situation, we eliminate the masking in the conventional self-attention mechanism. This approach preserves the valuable information in the learning sequence after pre-training while avoiding the introduction of unnecessary noise.
- Our experimental results illustrate that Self-KT surpasses state-of-the-art models on three public datasets, demonstrating improved AUC values of at least 2.9%.

II. Related Work

This section focuses on addressing the limitations of pre-trained models in extracting high-level semantic information between questions and skills in the context of Knowledge Tracing (KT), as well as the suboptimal performance of fine-tuned models based on attention mechanisms in KT. Firstly, we provide a comprehensive review of previous knowledge tracing models. Subsequently, we introduce a novel KT model that effectively combines pre-training with attention mechanisms.

A. Knowledge Tracing

In recent years, due to the development of deep neural networks, they have been used in many KT models [11]–

[13]. DKT first introduced deep learning into KT [5], [14], employing RNN or LSTM [15], [16] for modeling students' knowledge status. To fully capture the relationship between questions and skills, GKT [17] and GIKT [18] proposed using Graph Neural Networks (GNN) [19], [20] to obtain higher-order embeddings of questions from the relationships between skills. DAGKT [1] enriched the representation of question difficulty and the number of student attempts based on GIKT and is one of the most advanced models in the field of graph-based KT.

B. Pre-training

In order to capture the differences between specific questions and to maximize the extraction of rich high-level information from questions and skills, Pre-training Embeddings via Bipartite Graph (PEBG) proposed a method of pre-training. The method represents questions and skills as a bipartite graph, considering explicit question-skill relationships and implicit skill similarity and question similarity. It also integrates question difficulty as supplementary information. The Multi-Factors Aware Dual-Attentional model (MF-DAKT) [8] introduced a pre-training method that incorporates question relationships and difficulties into question representations using factor analysis, and then applies dual-attention for knowledge tracing. Although PEBG and MF-DAKT have developed different pre-training methods, each has its own emphasis on difficulty representation. For example, PEBG uses the first response time and type of question to describe question difficulty, whereas MF-DAKT uses question error rates. Additionally, their feature fusion cannot dynamically adjust the weights of different features based on the context. This may result in certain feature information being overly ignored or overemphasized, thereby affecting the overall effect of pre-training.

C. Attention

With the rise of Transformer [21], some studies [22] have given greater consideration to the use of attention mechanisms for knowledge tracing. The Self-Attentive Knowledge Tracing (SAKT) [6] introduced a self-attention mechanism to obtain weights for students' historical performance, aiming to address the issue of sparsity in real-world data. The Relation-aware self-attention model for Knowledge Tracing (RKT) [23] enhances KT's self-attention mechanism using question context information. However, a pure self-attention mechanism often fails to preserve the order of input sequences, making it difficult to effectively stack multiple attention layers. This limits the model's expressive capacity and its ability to model complex relationships. To overcome these issues, the Attentive Knowledge Tracing (AKT) [24] uses an encoder-decoder architecture to preserve the order of input sequences. The Separated Self-Attentive Neural Knowledge Tracing (SAINT) [25] model, based on Transformer, inputs the question and response embedding sequences separately into the encoder and decoder. This separation of inputs allows the model to stack attention layers multiple times, thereby improving the AUC.

In our work, to model the dependency relationships between different questions within a student's answer sequence, we improved pre-training difficulty representation and feature fusion methods. To make the most of the semantic information post-pre-training, we abandoned the position encoding and masking based on encoder self-attention, aiming to overcome the limitations of past knowledge tracing models and enhance their performance.

III. Methodology

In this section, we present a comprehensive description of our model, Self-KT, as depicted in Fig. 2. To tackle the challenges associated with representing question difficulty and feature fusion, as well as capturing the dynamic changes in student response sequences, we propose a comprehensive approach for computing question difficulty. Additionally, we introduce a novel feature fusion method to capture interactions among higher-order features. Lastly, we utilize a deep self-attention mechanism to take into account factors related to students' insightful modifications in their answers.

A. Problem Definition

Taking into account a student's prior question interactions prior to time step $t-1$, our objective is to predict the likelihood of the student answering a new question q correctly at time step t. The student's question records can be represented as $\boldsymbol{I} = (q_1, r_1), ..., (q_{t-1}, r_{t-1})$, where r_i denotes the correctness of the student's answer to question q_i at time step i. Specifically, $r_i \in \{0, 1\}$, where 1 indicates a correct response and 0 denotes an incorrect response. As such, the probability of the student answering the question i correctly can be expressed as $\boldsymbol{P}(r_t = 1|q_t, I)$.

Furthermore, the relationship between questions and skills can be divided into three types: the relationship between questions and skills, the similarity between questions, and the similarity between skills. These three types of relationships can be naturally represented as three bipartite graphs: $\boldsymbol{G}_{qs} = (Q, S, R)$, $\boldsymbol{G}_{qq} = (Q, Q, R^Q)$, $\boldsymbol{G}_{ss} = (S, S, R^S)$, where $\boldsymbol{Q} = \{q_i\}_{i=1}^{N_q}$ is the set of questions, $\boldsymbol{S} = \{s_i\}_{j=1}^{N_s}$ is the set of skills, N_q and N_s are the number of questions and skills, respectively. $\boldsymbol{R} = [r_{ij}] \in \{0, 1\}$ is a binary adjacency matrix. If there is an edge between the first two elements, then $r_{ij} = 1$, otherwise, $r_{ij} = 0$.

B. Input features

As shown in the input features of Fig. 2, we use question vertices, skill vertices, and question difficulty features as inputs. The vertex features are randomly initialized and updated during the pre-training phase, which learns a linear mapping from one-hot encoding to continuous features.

In our study, we consider the average first response time, the average number of attempts, and the question error rate among students as indicators of question difficulty. These metrics collectively provide a comprehensive understanding of the complexity of the questions. To analyze the relationship between questions and skills, we utilize the inner product

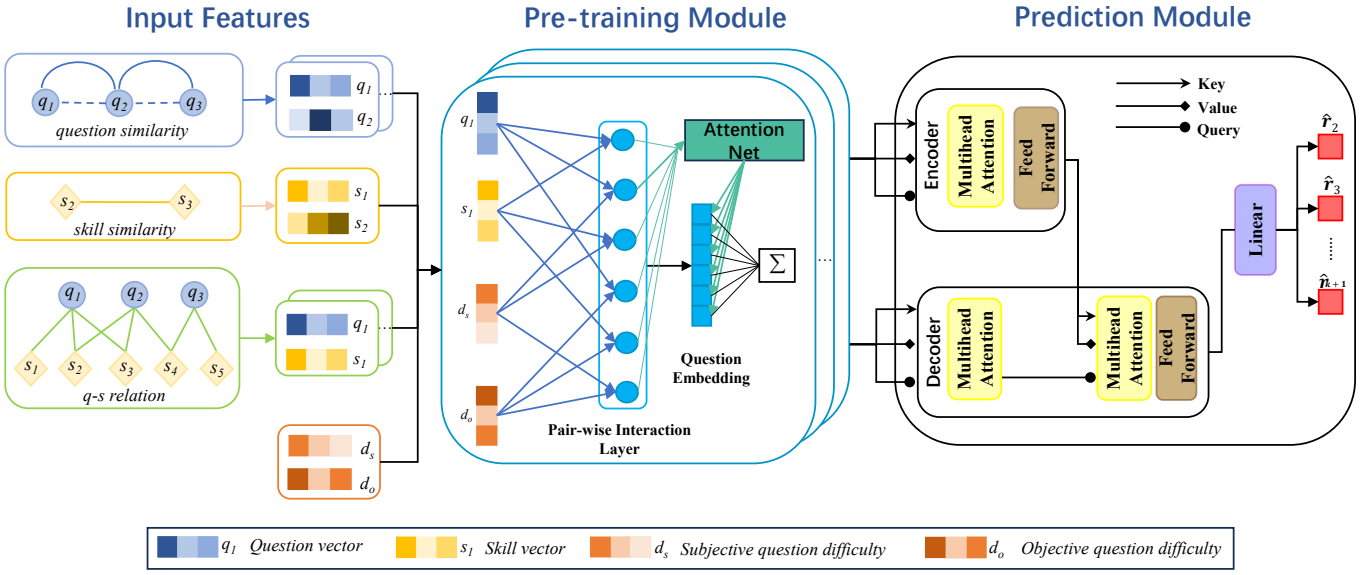


Fig. 2: The framework of Self-KT, including three modules: input features (detailed in Section III B), pre-training module (detailed in Section III C), and prediction module (detailed in Section III D).

in the embedding space. This approach effectively captures the proximity between the question and skill vertices. Furthermore, we estimate the similarities between questions and skills. To regulate the question-skill relationship and ensure appropriate similarities among questions, we employ the cross-entropy function $L_1(Q, S)$, $L_2(Q)$, and $L_3(S)$ [7].

C. Pre-training Process

The question q consists of vertex features q and attribute features f. we first use a linear layer parameterized by w_a to map the attribute features f to a low-dimensional feature representation, which is denoted as $a \epsilon \mathbb{R}^{d_v}$. Let S' represent the average representation of the vertex features for q-related skills.

We use vertex feature q, the average skill feature S', and the attribute features a to generate the linear information Z and the quadratic information P for the question q. Specifically,

$$Z = (x_1, x_2, x_3) \triangleq (q, s', a), \tag{1}$$

$$P =< x_i, x_j >, \tag{2}$$

where P is defined as a two-by-two feature interaction. Then we introduce an attention-based pooling layer, which can transform these two information matrices to signal vectors l_z and l_p. We propose to employ the attention mechanism on feature interactions by performing a weighted sum on the interacted vectors:

$$l_z{}^{(k)} = W_z^{(k)} \odot Z = \sum_{i=1}^{3} \sum_{j=1}^{d_v} (w_z^{(k)})_{ij} x_{ij}, \tag{3}$$

$$l_p{}^{(k)} = \sum_{i=1}^{3} \sum_{j=1}^{3} a_{ij} (w_p^{(k)})_{ij} x_i x_j, \tag{4}$$

$k \in [1, ...d]$, and $\odot$ denotes operations that firstly elementwise multiplication is applied to two matrices, then the multiplication result is summed up to a scalar. d is the transform dimension of l_z and l_p. $\mathrm{W}_z^{(k)}$ and $\mathrm{W}_p^{(k)}$ are the weights in the pair-wise interaction layer. And a_{ij} is the attention score for feature interaction $\mathrm{W}_p^{(k)}$, which can be interpreted as the importance of $\mathrm{W}_p^{(k)}$ in predicting the target. Formally, the attention network is defined as:

$$a'_{ij} = \mathrm{h}^T ReLU(\mathrm{W}(\mathrm{W}_p^{(R)})_{ij} x_i x_j + b), \tag{5}$$

$$a_{ij} = \frac{\exp a'_{ij}}{\sum_{(i,j) \in \mathcal{R}_x} \exp(a'_{ij})}, \tag{6}$$

where $\mathrm{W} \in \mathbb{R}^{3\times3}$, $\mathrm{b} \in \mathbb{R}^{3\times3}$, $\mathrm{h} \in \mathbb{R}^{3\times3}$ are model parameters, and t denotes the hidden layer size of the attention network.

Then, we can calculate the embedding of question q, which is denoted as e:

$$e = ReLU(l_z + l_p + b'), \tag{7}$$

where l_z, l_p and the bias vetor $b \in \mathbb{R}^d$, and $l_z = (l_z^{(1)}, l_z^{(2)}, ...l_z^{(d)})$, $l_p = (l_p^{(1)}, l_p^{(2)}, ...l_p^{(d)})$. The activation function is rectified linear unit (ReLU), defined as ReLU(x) = max(0, x).

we use a linear layer to map the activation e_i to a difficulty approximation $\hat{d}_i = w_d^T e_i + b_d$ where w_d and b_d are network parameters. We use the question difficulty d_i as the auxiliary target, and design the following loss function L4 to measure the difficulty approximation error:

$$L_4(Q, S, \theta) = \sum_{i=1}^{|Q|} (d_i - \hat{d}_i)^2, \tag{8}$$

where θ denotes all the parameters in the network, i.e., $\theta = \{w_a, \mathrm{W}_z, \mathrm{W}_p, w_d, b, b', b_d\}$. we combine all the loss functions to form a joint optimization framework, namely, we solve:

$$\min_{Q,S,\theta} \lambda(L_1(Q,S) + L_2(Q) + L_3(S)) + (1-\lambda)L_4(Q,S,\theta), \tag{9}$$

where λ is a coefficient to control the trade-off between bipartite graph constraints and difficulty constraint.

D. Deep Self-attentive Encoder-Decoder

To obtain input from pre-training, there are two embedding matrices: $\boldsymbol{Q} \in \mathbb{R}^{q*d}$, $\boldsymbol{I} \in \mathbb{R}^{2q*d}$, where q is the total number of question, and d is the latent dimension. Then we use two projection matrices $\mathbf{W}^I \in \mathbb{R}^{d*d}$, $\mathbf{W}^E \in \mathbb{R}^{d*d}$ to project the respective vectors to different space linearly. The output of embedding layer are embedded vectors $\boldsymbol{Q} \in \mathbb{R}^{n*d}$, $\boldsymbol{I} \in \mathbb{R}^{n*d}$, where n is the number of the sequence length.

Encoder. Formally, the encoder can be presented as:

$$\begin{cases} m = \text{LayerNorm}(x + \text{Multihead}(x,x,x)), \\ z = \text{LayerNorm}(m + \text{FFN}\ (m)) \\ \text{Encoder}\,(x) = z \end{cases} \tag{10}$$

In the encoder, the input sequence x consists of question embedding sequences, and each subsequent layer uses the preceding layer's feedforward output as input. Layer normalization and residual connections are applied after each sub-layer.

Decoder. Similar to the encoder, our multi-head attention layer will be used twice with the same weight in the calculation. For the second time, the keys and values are the same vectors obtained from the encoder. The computation of the decoder layer can be summarized as follows:

$$\begin{cases} m_1 = \text{LayerNorm}(x + \text{Multihead}(x,x,x)), \\ m_2 = \text{LayerNorm}(m_1 + \text{Multihead}(m_1,x',x')) \\ z = \text{LayerNorm}(m_2 + \text{FFN}(m_2)), \\ \text{Decoder}\,(x,x') = z \end{cases} \tag{11}$$

Network Training. The training objective adopted in this paper is to minimize the negative log-likelihood loss of the observed student response sequences in the model.

IV. Experiments

In this section, we validated our model on several real-world datasets and conducted ablation studies and visualizations to assess the impact of each key component in the Self-KT architecture.

A. Dataset

We evaluate the performance of the Self-KT model using three commonly used publicly available datasets [26] in the field of Knowledge Tracing.

- **Assistment2009**[1] **and Assistment2012**[2]**.** The dataset used in this study was collected from the ASSISTments online education platform [27]. Assistment2009 dataset consists of 3,841 students, 123 skills, 15,911 questions, and 190,320 records. Assistment2012 dataset consists of 1,867,167 records from 27,405 students, encompassing 265 skills and 47,104 questions.
- **Ednet**[3]**.** This dataset is collected by [28]. There are in total 188 skills and 10,779 questions. The largest number of skills underlying one question is 6.

To reduce noise, we excluded skill tags that were unnamed or dummies for three datasets. Moreover, we only utilized the sequence of students whose history records were longer than 3, as history question records that are too short would be meaningless to predict. The statistics of the three datasets are presented in Table I.

TABLE I: The statistics of datasets

	Assistment2009	Assistment2012	Ednet
student	3,834	27,405	5,002
questions	15,911	47,104	10,779
skills	123	265	188
records	190,320	1,867,167	222,141
skill per question	1.207	1.000	2.210
records per question	11.96	39.64	20.57

B. Baselines

To demonstrate the effectiveness of our model and show the improvement over existing deep KT models, we compare the predictive performance of state-of-the-art deep KT models.

1) Deep Sequential Models

- **DKT.** It is the first model to use deep learning techniques in a KT model, using a single layer of LSTM to predict student performance.
- **DKVMN.** The model uses a key-value memory network to store and update students' understanding of each skill.
- **DKT-Q.** It is an extension of DKT that utilizes questions as input to the model.
- **DKVMN-Q.** It is an extension of DKVMN that directly uses questions for prediction.
- **AKT.** It is a context-aware model that uses a novel monotonic attention mechanism to connect a learner's future responses to evaluation questions with their past responses.
- **SAKT.** It uses a self-attention mechanism to simulate the relationships between interactions at different time steps.
- **GKT.** It converts the knowledge structure into a graph, enabling the model to reformulate the knowledge tracking task as a time-series node-level classification problem in GNN.

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

[2]https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect

[3]https://github.com/riiid/ednet

- **GIKT.** It utilizes a graph convolutional network to substantially incorporate question-skill correlations via embedding propagation.

2) Pre-training Models

- **PEBG-DKT.** It is the first method to pre-training question embeddings, and it represents the relationship between questions and skills using a bipartite graph.
- **MF-DAKT.** It proposes a new student-related factor that tracks recent attempts by the student on related skill to highlight the influence of recent practice.

In this paper, we employed two commonly used metrics for KT: AUC (Area Under the ROC curve) and ACC (Accuracy) [8].

We utilize $PyTorch$ and the Adam algorithm to optimize our model, setting the batch size to 128, the question embedding dimension d = 128, and the learning rate for all three datasets to 0.001. To alleviate overfitting, we also introduce a dropout of 0.2. We divide each dataset into 80% for training and 20% for testing.

C. Prediction Performance

In this section, Table II presents the predictive performance of all compared models, with the highest performance indicated in bold. We can observe that the proposed Self-KT model achieves the highest AUC and ACC on all three datasets. On the Assistment2009 dataset, the AUC of the Self-KT model reaches 86.18%. Among previous non-pre-trained models, AKT and GIKT, as the most advanced models, achieved AUC values of 80.46% and 75.85% respectively. Among other models using pre-training, our model on average outperforms the PEBG-DKT and MF-DAKT models by 3.85%, which have AUCs of 82.45% and 82.21% respectively. On the Assistment2012 dataset, the results indicate that Self-KT achieved an AUC of 79.97%, representing an average increase of 3.92% compared to the AUCs of 76.14% for PEBG-DKT and 77.10% for MF-DAKT. On the Ednet dataset, the AUC of Self-KT was 4.97% higher than the average of PEBG-DKT and MF-DAKT.

Table II results indicate that the pre-trained PEBG-DKT and MF-DAKT models perform similarly. Their performance significantly surpasses all non-pre-trained models, suggesting that pre-training models can address the issues caused by sparse question interactions and incomplete associations between skills. However, their performance still falls short compared to our proposed model. We attribute this to the insufficient representation of question difficulty during the pre-training phase and the neglect of the impact of feature fusion throughout pre-training. Considering the mutual inspiration and correction factors between questions during the prediction phase plays a crucial role in the model.

Training set size. The AKT and MF-DAKT models suggest that an excessive number of questions can lead to the issue of overparameterization. To verify whether our proposed Self-KT model would also face this issue, we conducted experiments on two different types of datasets. We divided the datasets into 80%, 60%, 40%, and 20% based on the number of samples to observe the changes in model performance. The experimental results are shown in Fig. 3. Regardless of whether it is the pre-trained Self-KT, PEBG-DKT, and MF-DAKT models, or the AKT model based on IRT, their performance fluctuations are relatively small. This indicates that the pre-training technology of Self-KT and the other three models can effectively alleviate the problems brought about by over-parameterization. Furthermore, regardless of how the size of the training set changes, the Self-KT model can achieve excellent performance, which further demonstrates its strong adaptability and robustness.

TABLE II: The Results over Three Datasets

Model	Assistment2009		Assistment2012		Ednet	
	AUC	ACC	AUC	ACC	AUC	ACC
DKT	0.7335	0.7514	0.6656	0.7066	0.6843	0.6566
DKVMN	0.7456	0.7527	0.7110	0.7184	0.6867	0.6509
DKT-Q	0.7145	0.7318	0.6537	0.6841	0.6932	0.6759
DKVMN-Q	0.7468	0.7544	0.7011	0.7046	0.6956	0.6681
AKT	0.8046	0.7568	0.7565	0.7357	0.7322	0.7151
SAKT	0.7297	0.7066	0.7360	0.7011	0.7652	0.7286
GKT	0.7446	0.7581	0.7554	0.7325	0.6170	0.6120
GIKT	0.7585	0.7438	0.7601	0.7354	0.6320	0.6368
PEBG-DKT	0.8245	0.7630	0.7614	0.7426	0.7715	0.7129
MF-DAKT	0.8321	0.7688	0.7710	0.7471	0.7402	0.7156
Self-KT	**0.8618**	**0.7876**	**0.7997**	**0.7604**	**0.8185**	**0.7407**

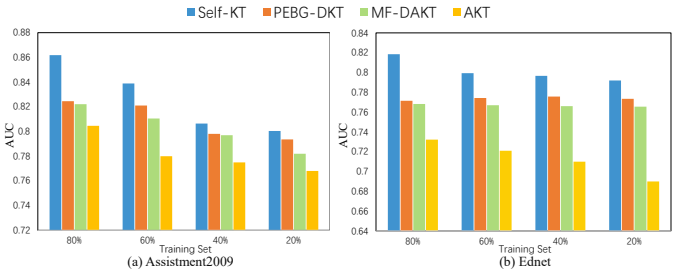


Fig. 3: Performance on different size training sets.

D. Ablation Study

In this section, we conducted two ablation experiments to validate the effectiveness of the components in our model. In the first experiment, we performed ablations on the features and their fusion, and the results are presented in Table III. Details of the experimental settings are listed below:

- **REQ** (Remove the Error Rate of the Questions) without considering the difficulty of question error rates.
- **RAP** (Replace AFM Layer with Product Layer) using product layer as the pre-trained feature fusion.
- **RAF** (Replace AFM Layer with Fully Connected Layer) using fully connected layers instead of pre-trained feature fusion methods.

- **RAL** (Remove AFM Layer) without considering pre-trained feature fusion.

TABLE III: Ablation Study of Pre-training

Model	Assistment2009		Assistment2012		Ednet	
	AUC	ACC	AUC	ACC	AUC	ACC
Self-KT	**0.8618**	**0.7876**	**0.7997**	**0.7604**	**0.8185**	**0.7407**
REQ	0.8504	0.7863	0.7895	0.7439	0.8075	0.7343
RAP	0.8591	0.7870	0.7938	0.7424	0.8107	0.7366
RAF	0.8477	0.7788	0.7920	0.7542	0.7951	0.7115
RAL	0.8524	0.7739	0.7922	0.7477	0.7963	0.7159

In addition to the aforementioned modifications, no other changes were made to the model and experimental setup. As indicated in Table III, Self-KT outperformed all other models on all datasets. The performance of REQ would worsen, especially on the Ednet dataset, if the question difficulty feature represented by the error rate of the question is removed. This is due to the larger proportion of student questions in the Ednet dataset, making the representation of question difficulty crucial for students' final performance. From the analysis of RAF and RAL, it can be observed that the absence of feature fusion negatively impacts the model's performance. In comparison, RAP, which utilizes the product layer as the feature fusion layer, exhibits better performance. Furthermore, when comparing RAP and Self-KT, it is evident that the use of AFM yields superior performance. This further confirms that AFM is capable of capturing the uniqueness among various features and dynamically allocating weights to them.

In the second set of ablation experiments, we conducted ablations on the model structure or parameters, and the results are presented in Table IV. The experimental settings are listed as follows:

- **AM** (Add mask) using traditional attention mechanisms with masks.
- **RED** (Remove the Encoder-Decoder part) direct stacking of multi-headed attention without considering the encoder-decoder structure.
- **RMD** (Reduce one multi-head attention from the decoder) without considering skip connections, using either serial connections or multi-head attention in the encoder and decoder.
- **RPP** (Replace the pre-training part with PEBG) directly combining the PEBG pre-trained model with the fine-tuning model.
- **RD** (Remove dropout) without considering the impact of dropout on generalization.

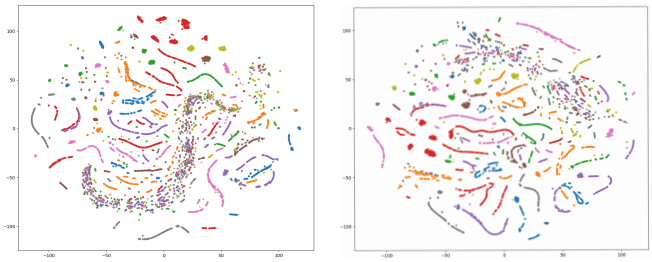
Similarly, aside from the experimental modifications mentioned previously, all other aspects of the models and experimental setup used in this study remained unchanged. As shown in Table IV, The introduction of masking often disrupts the partially existing relationships between pre-training questions, leading to a decline in model performance. The RED indicates

TABLE IV: Ablation Study of Key Components

Model	Assistment2009		Assistment2012		Ednet	
	AUC	ACC	AUC	ACC	AUC	ACC
Self-KT	**0.8618**	**0.7876**	**0.7997**	**0.7604**	**0.8185**	**0.7407**
AM	0.8291	0.7564	0.7677	0.7277	0.7716	0.6794
RED	0.8476	0.7801	0.7941	0.7412	0.8111	0.7303
RMD	0.8492	0.7736	0.7852	0.7483	0.7967	0.7147
RPP	0.8546	0.7818	0.7686	0.7235	0.8061	0.7305
RAL	0.8568	0.7812	0.7912	0.7350	0.8109	0.7325

that when the encoder-decoder module is eliminated, it hinders the acquisition of low-dimensional vector representations, impeding the effective handling of relationships between input and output sequences. Furthermore, when we only alter the structure of the decoder, for instance by removing multi-head attention, the performance of RMD deteriorates, becoming worse than that of Self-KT. We speculate that a single multi-head attention layer in the decoder may not be sufficient to effectively capture different levels of features and information in the input sequence. RPP implies that different pre-trained models have varying degrees of impact on model performance, further validating that the effectiveness of the pre-training part in Self-KT is greater than the current state-of-the-art pre-training model PEBG. The RD analysis shows that the effect of eliminating dropout on the model's generalization ability is consistent across all three datasets, indicating that the impact of dropout is similar across different datasets.

E. Visualization



(a) Pre-training (b) Self-KT

Fig. 4: A comparison was made between the question embeddings learned by deep knowledge tracing models at the question level using the Assistment2009 dataset.

On the Assistment2009 dataset, we employed t-SNE [29] to project the multidimensional item embeddings generated by Self-KT pre-training and the entire model optimization process onto a two-dimensional space. During this process, questions involving the same skill were labeled with the same color. As shown in Fig. 4(a), the pre-trained question representations exhibit a good structure. After further fine-tuning by Self-KT, the question representations are projected

onto Fig. 4(b), where we observe that the representations still maintain their relationships and even exhibit a more refined structure compared to Fig. 4(a). We believe that students' practice records can assist pre-trained question representations in capturing the relationships between questions during the fine-tuning process.

V. Conclusion

In this study, we propose new methods for representing question difficulty and feature fusion technique to address the limitations of current pre-training methods, specifically tailored for pre-training applications. To capture the authentic dynamic changes in the student's response process, our model adopts a multi-head attention encoder and decoder architecture. Empirical research conducted on real-world datasets demonstrates that our proposed Self-KT model significantly improves the performance of existing deep knowledge tracking models.

Acknowledgment

This work was supported in part by the Key Project of Beijing Higher Education Association in 2023 under Grant No. ZD202311, Beijing Information Science and Technology University Teaching Reform Project under Grant No. 2024JGYB23.

References

[1] R. Luo, F. Liu, W. Liang, Y. Zhang, C. Bu, and X. Hu, "Dagkt: Difficulty and attempts boosted graph-based knowledge tracing," *arXiv preprint arXiv:2210.15470*, 2022.

[2] G. Abdelrahman and Q. Wang, "Knowledge tracing with sequential key-value memory networks," in *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 2019, pp. 175–184.

[3] C. Wang, S. Sahebi, S. Zhao, P. Brusilovsky, and L. O. Moraes, "Knowledge tracing for complex problem solving: Granular rank-based tensor factorization," in *Proceedings of the 29th ACM conference on user modeling, adaptation and personalization*, 2021, pp. 179–188.

[4] C. Wang, W. Ma, M. Zhang, C. Lv, and S. Ma, "Temporal cross-effects in knowledge tracing," in *WSDM '21: The Fourteenth ACM International Conference on Web Search and Data Mining*, 2021.

[5] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 765–774.

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

[7] Y. Liu, Y. Yang, X. Chen, J. Shen, H. Zhang, and Y. Yu, "Improving knowledge tracing via pre-training question embeddings," *arXiv preprint arXiv:2012.05031*, 2020.

[8] M. Zhang, X. Zhu, C. Zhang, Y. Ji, F. Pan, and C. Yin, "Multi-factors aware dual-attentional knowledge tracing," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2588–2597.

[9] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," *arXiv preprint arXiv:1708.04617*, 2017.

[10] J. Zeng, Q. Zhang, N. Xie, and B. Yang, "Application of deep self-attention in knowledge tracing," *arXiv preprint arXiv:2105.07909*, 2021.

[11] K. Nagatani, Q. Zhang, M. Sato, Y. Y. Chen, and T. Ohkuma, "Augmenting knowledge tracing by considering forgetting behavior," in *The World Wide Web Conference*, 2019.

[12] T. Long, J. Qin, J. Shen, W. Zhang, W. Xia, R. Tang, X. He, and Y. Yu, "Improving knowledge tracing with collaborative information," in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 599–607.

[13] Y. Zhou, X. Li, Y. Cao, X. Zhao, Q. Ye, and J. Lv, "Lana: towards personalized deep knowledge tracing through distinguishable interactive sequences," *arXiv preprint arXiv:2105.06266*, 2021.

[14] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *Advances in neural information processing systems*, vol. 28, 2015.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[17] H. Nakagawa, Y. Iwasawa, and Y. Matsuo, "Graph-based knowledge tracing: modeling student proficiency using graph neural network," in *IEEE/WIC/ACM International Conference on Web Intelligence*, 2019, pp. 156–163.

[18] Y. Yang, J. Shen, Y. Qu, Y. Liu, K. Wang, Y. Zhu, W. Zhang, and Y. Yu, "Gikt: a graph-based interaction model for knowledge tracing," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*. Springer, 2021, pp. 299–315.

[19] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[20] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, and A. Mian, "Bi-clkt: Bi-graph contrastive learning based knowledge tracing," 2022.

[21] A. Waswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[22] Y. Su, Q. Liu, Q. Liu, Z. Huang, Y. Yin, E. Chen, C. Ding, S. Wei, and G. Hu, "Exercise-enhanced sequential modeling for student performance prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[23] S. Pandey and J. Srivastava, "Rkt: relation-aware self-attention for knowledge tracing," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1205–1214.

[24] A. Ghosh, N. Heffernan, and A. S. Lan, "Context-aware attentive knowledge tracing," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 2330–2339.

[25] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo, "Towards an appropriate query, key, and value computation for knowledge tracing," in *Proceedings of the seventh ACM conference on learning@ scale*, 2020, pp. 341–344.

[26] G. Abdelrahman, Q. Wang, and B. Nunes, "Knowledge tracing: A survey," *ACM Comput. Surv.*, vol. 55, no. 11, feb 2023. [Online]. Available: https://doi.org/10.1145/3569576

[27] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge with an online system that tutors as it assesses," *User Modeling and User-Adapted Interaction*, vol. 19, no. 3, pp. 243–266, 2009.

[28] Y. Choi, Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo, "Ednet: A large-scale hierarchical dataset in education," in *Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part II 21*. Springer, 2020, pp. 69–73.

[29] G. Hinton and L. van der Maaten, "Visualizing data using t-sne journal of machine learning research," 2008.