AdvKT: An Adversarial Multi-Step Training Framework for Knowledge Tracing

Lingyue Fu[1], Ting Long[2], Jianghao Lin[1], Wei Xia[4], Xinyi Dai[3], Ruiming Tang[3], Yasheng Wang[3], Weinan Zhang[1](✉), and Yong Yu[1](✉)

[1] Shanghai Jiao Tong University
{fulingyue, chiangel, wnzhang, yyu}@sjtu.edu.cn
[2] Jilin University longting@jlu.edu.cn
[3] Huawei Noah's Ark Lab
{daixinyi5, tangruiming, wangyasheng}@huawei.com
[4] www.imxwell.com xwell.xia@gmail.com

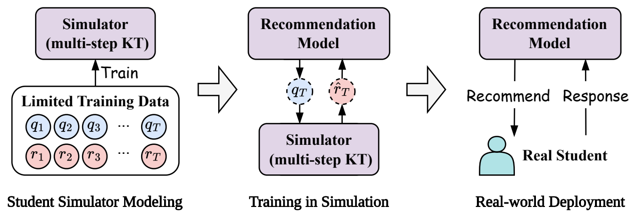
Abstract. Knowledge Tracing (KT) monitors students' knowledge states and simulates their responses to question sequences. Existing KT models typically follow a single-step training paradigm, which leads to discrepancies with the multi-step inference process required in real-world simulations, resulting in significant error accumulation. This accumulation of error, coupled with the issue of data sparsity, can substantially degrade the performance of recommendation models in the intelligent tutoring systems. To address these challenges, we propose a novel Adversarial Multi-Step Training Framework for Knowledge Tracing (AdvKT), which, for the first time, focuses on the multi-step KT task. More specifically, AdvKT leverages adversarial learning paradigm involving a generator and a discriminator. The generator mimics high-reward responses, effectively reducing error accumulation across multiple steps, while the discriminator provides feedback to generate synthetic data. Additionally, we design specialized data augmentation techniques to enrich the training data with realistic variations, ensuring that the model generalizes well even in scenarios with sparse data. Experiments conducted on four real-world datasets demonstrate the superiority of AdvKT over existing KT models, showcasing its ability to address both error accumulation and data sparsity issues effectively.

Keywords: Knowledge Tracing · Educational Data Mining · Student Simulator

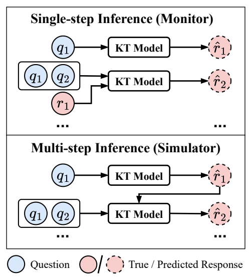
1 Introduction

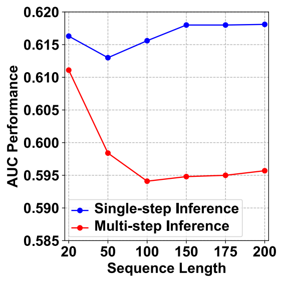
Intelligent tutoring systems (ITS), such as Massive Online Open Courses (MOOCs), aim to help students learn more efficiently through learning path recommendation and question recommendation. Knowledge Tracing (KT) is a vital component of ITS, which estimates the knowledge state by predicting the student responses, i.e., whether a student can answer each question correctly or not. KT models are used in two ways: as monitors to provide human instructors with insights into students' mastery levels, and as *student simulators* to supply reward signals for recommendation models.

Existing KT models [2,27,8,20,17] show promising performance as monitors of students' learning progress. By taking students' real-time responses to various questions



(a) Application of the student simulator.





(b) Inference Comparison. (c) AUC Comparison.

Fig. 1: (a) Illustration of the students simulator's application and the comparison between single-step and multi-step inference KT. (b) Comparison of single-step inference and multi-step inference (simulator) and the primary application of the simulator. Input of single-step inference: historical question and ground-truth response. Input of multi-step inference: historical question and predicted response. (c) Performance comparison of the single-step inference and multi-step inference with a single-step training approach. With increased sequence length, errors accumulate gradually under the multi-step inference setting.

as input, these models estimate the knowledge states of students and make predictions for the next questions. The predictions can be a reference for human instructors, helping them identify areas where students may need additional support or challenge.

KT models are more frequently employed in ITS as *student simulators* for question recommendation models [19,9,16]. As illustrated in Figure 1(a), after the recommendation model generates a sequence of questions, the pre-trained simulator (KT model) provides simulated student responses, which serve as rewards to optimize the recommendation model. The objective of the recommendation model is to maximize the reward provided by the student simulator. In this setting, the accuracy of the simulation has a significant impact on the effectiveness of the recommendation model in the real-world deployment phase.

Despite their widespread use, existing KT models largely overlook their role as student simulators, leading to a gap between *single-step training* and *multi-step inference*. Typically, these models follow a single-step training paradigm, where they are optimized to predict a student's response for only the next question in a sequence, assuming the ground-truth response history is always accessible. In real-world simulations, however, the simulator must predict responses for an entire recommended question sequence at once. As illustrated in Figure 1(b), during multi-step inference, the simulator iteratively generates responses based on its previous predictions, as ground-truth history is unavailable.

The discrepancy between single-step training and multi-step inference results in significant *error accumulation* over longer sequences, adversely affecting the recommendation model's performance. As shown in Figure 1(c), we train EERNNA [30] using a single-step training approach and evaluate it under both single-step and multi-step inference settings. In the single-step setting, where ground-truth history is available, the model performs stably across varying sequence lengths, as input sequences remain error-free. In contrast, under the multi-step inference setting, prediction errors gradually accumulate as the sequence length increases, leading to a substantial degradation in performance. This discrepancy highlights the need for KT models that better align with the multi-step inference nature of real-world simulations.

Furthermore, current ITS suffer from the data sparsity problem. Existing KT models are generally supervised by optimizing the binary cross-entropy (BCE) loss and, therefore, minimizing the KL divergence between the model and data distributions, which casts heavy demands on the quantity of training data. However, in real-world ITS, the number of students in each class is typically limited, and each student exercises a finite number of questions. For example, in ASSIST09 dataset[5], there are only 2,661 students with 165,455 interactions, which contains 14,083 questions. The interactive data are sparse with respect to the whole space of question sequences, thus it is difficult for KT models to capture the students' learning patterns precisely.

To conquer the above challenges, we propose a novel Adversarial Multi-Step Training Framework for Knowledge Tracing, named **AdvKT**. To the best of our knowledge, this is the first work to formalize the multi-step KT task and the first to introduce adversarial learning to the KT task. AdvKT enables models to effectively handle the iterative nature of multi-step simulations by leveraging adversarial learning to diversify training sequences, thereby mitigating error propagation during inference. Additionally, we propose specialized data augmentation techniques to enrich training data with realistic variations, ensuring the model generalizes effectively even in sparse data scenarios.

To sum up, our contributions are summarized as follows:

- We propose a novel Adversarial Multi-Step Training Framework for the KT task (*i.e.*, **AdvKT**) to address the challenge of error accumulation in multi-step inference. To the best of our knowledge, this is the first work that focuses on the nature of multi-step prediction in the KT task.
- We design tailored data augmentation techniques specifically for KT tasks and incorporate them into AdvKT, thereby reducing the prediction bias caused by the data sparsity problem and uneven distributions.
- Extensive experiments on four real-world datasets demonstrate that AdvKT achieves state-of-the-art performance under the multi-step inference setting, as well as superior capabilities to tackle the data sparsity.

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

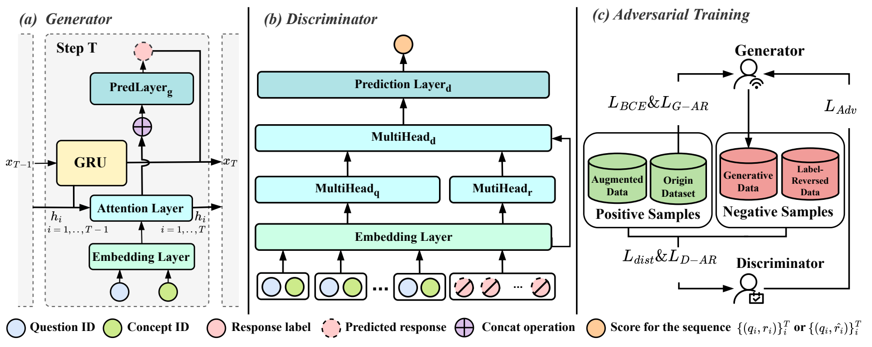


Fig. 2: The overview of the proposed AdvKT framework.

2 Problem Formulation

In ITS, let $\mathcal{S}$ represent the set of students, $\mathcal{Q}$ the set of questions, and $\mathcal{C}$ the set of concepts. Each question $q \in \mathcal{Q}$ is associated with a subset of concepts from the set $\mathcal{C}$, denoted as $\mathcal{C}^q = \{c_1^q, \ldots, c_k^q\} \subset \mathcal{C}$. The student's learning sequence is recorded as $s = [(q_i, r_i)]_{i=1}^T$, where $q_i \in \mathcal{Q}$ is the question, and $r_i \in 0, 1$ indicates whether the student answered question q_i correctly ($r_i = 1$ for correct, and $r_i = 0$ for incorrect).

Previous KT models typically follow a single-step training and inference paradigm, where the ground-truth responses to previous questions are used to predict the next response. The single-step prediction at step T can be formulated as:

$$\hat{r}_T = p\left(r_T = 1 | [(q_t, r_t)]_{t=1}^{T-1}, q_T\right). \tag{1}$$

In contrast to existing KT models, we focus on a multi-step prediction paradigm, which is more in line with real-world applications. Specifically, this approach involves predicting the student's response r_T using previously predicted responses $[\hat{r}_t]_{t=1}^{T-1}$, which can be expressed as:

$$\hat{r}_T = p\left(r_T = 1 | [(q_t, \hat{r}_t)]_{t=1}^{T-1}, q_T\right). \tag{2}$$

3 Methodology

In this section, we introduce the framework of AdvKT, as dipicted in Figure 2. AdvKT consists of two main components: a generator that predicts the student's responses for a given question sequence, and a discriminator that distinguishes between real and fake data. We employ adversarial training over the generator and the discriminator with data augmentation.

3.1 Embedding Layer

To project discrete IDs to dense representations, we create a question embedding matrix $\mathbf{Emb}^Q \in \mathbb{R}^{|\mathcal{Q}|\times d}$ and a concept embedding matrix $\mathbf{Emb}^C \in \mathbb{R}^{|\mathcal{C}|\times d}$, where d denotes the embedding size. Let $\mathbf{e}_q = \mathbf{Emb}_q^Q \in \mathbb{R}^d$ and $\mathbf{e}_c = \mathbf{Emb}_c^C \in \mathbb{R}^d$ be the embedding vectors of question q and concept c, respectively. Following [32], we represent question q by concatenating the question embedding with the mean aggregation of its corresponding concept embeddings:

$$\mathbf{v}_{c,q} = \frac{1}{|\mathcal{C}^q|} \sum\nolimits_{i=1}^{|\mathcal{C}^q|} \mathbf{e}_{c_i^q}, \mathbf{v}_q \quad = \mathbf{e}_q \oplus \mathbf{v}_{c,q}, \tag{3}$$

where $\oplus$ denotes the vector concatenation. Similarly, embedding vectors of position $o \in [1, T]$ and response $r \in \{0, 1\}$ are $\mathbf{e}_o = \mathbf{Emb}_o^O \in \mathbb{R}^d$ and $\mathbf{e}_r = \mathbf{Emb}_r^R \in \mathbb{R}^d$, respectively.

3.2 Generator

The generator $\mathcal{G}([(q_t, \hat{r}_t)]_{i=t}^{T-1}, q_T)$ predicts the student's response to question q_T based on historical questions with its own predicted responses. We utilize Gated Recurrent Unit (GRU) to represent students' learning states at step T

$$\mathbf{h}_T = \text{GRU}_g(\mathbf{x}_T, \mathbf{h}_{T-1}), \tag{4}$$

where $\mathbf{x}_T$ is the student interaction vector in step T which would be introduce later. According to [30], the student's current state is a weighted sum aggregation of all the historical states. Hence, we assign different levels of importance to the historical states by applying an attention layer:

$$\begin{aligned} \mathbf{Q} = \mathbf{v}_{q_T}, \mathbf{K} = \{\mathbf{v}_{q_1}, \cdots, \mathbf{v}_{q_{T-1}}\}, \mathbf{V} = \{\mathbf{h}_1, \cdots, \mathbf{h}_{T-1}\}, \\ \mathbf{a}_T = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}). \end{aligned} \tag{5}$$

Next, we apply a prediction layer to calculate the probability of answering question q_T correctly:

$$\hat{r}_T = \text{Sigmoid}\left(\text{PredLayer}_g(\mathbf{v}_{q_T} \oplus \mathbf{h}_{T-1} \oplus \mathbf{a}_T)\right), \tag{6}$$

where PredLayer is a two-layer fully connected network.

Finally, according to the predicted student response $\hat{r}_T$, we concatenate the question representation $\mathbf{v}_{q_T}$ and weighted historical state $\mathbf{a}_T$ to form the student interaction vector $\mathbf{x}_T$, which serves as the recurrent input for the next GRU process:

$$\mathbf{x}_T = \begin{cases} \mathbf{v}_{q_T} \oplus \mathbf{a}_T \oplus \mathbf{0_q} \oplus \mathbf{0_a}, \hat{r} \geq 0.5, \\ \mathbf{0_q} \oplus \mathbf{0_a} \oplus \mathbf{v}_{q_T} \oplus \mathbf{a}_T, \hat{r} < 0.5, \end{cases} \tag{7}$$

where $\mathbf{0_q}$ and $\mathbf{0_a}$ are zero vectors with the same size as the vector $\mathbf{v}_{q_T}$ and $\mathbf{a}_T$, respectively.

3.3 Discriminator

The discriminator $\mathcal{D}\left([(q_t, r_t)]_{t=1}^T\right)$ distinguishes whether the sequence is derived from real data (*i.e.*, positive) or fake data (*i.e.*, negative). The responses in the input sequence can be either ground-truth responses r_i or generated ones $\hat{r}_i$. For simplicity, hereinafter, we use r_i to denote the input responses. We adopt the classical multi-head attention architecture to encode the sequence, which can be formulated as:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= (\mathbf{head}_1 \oplus \cdots \oplus \mathbf{head}_h)\, \mathbf{W}^o, \\ \mathbf{head}_i &= \text{Attention}\left(\mathbf{W}_i^q \mathbf{Q}, \mathbf{W}_i^k \mathbf{K}, \mathbf{W}_i^v \mathbf{V}\right), \end{aligned} \tag{8}$$

where the number of heads h is a hyperparameter, $\mathbf{W}_i^q, \mathbf{W}_i^k, \mathbf{W}_i^v$ and $\mathbf{W}^o$ represent the learnable parameters.

The discriminator takes the question sequence $[q_i]_{i=1}^T$, response sequence $[r_i]_{i=1}^T$, and positions $[o_i]_{i=1}^T$ as inputs. We define the question-position matrix $\mathbf{v}_q$ and response-position matrix $\mathbf{v}_r$ as:

$$\mathbf{V}_q = \begin{pmatrix} \mathbf{e}_{q_1} \oplus \mathbf{e}_{o_1} \\ \mathbf{e}_{q_2} \oplus \mathbf{e}_{o_2} \\ \cdots \\ \mathbf{e}_{q_T} \oplus \mathbf{e}_{o_T} \end{pmatrix}, \ \mathbf{V}_r = \begin{pmatrix} \mathbf{e}_{r_1} \oplus \mathbf{e}_{o_1} \\ \mathbf{e}_{r_2} \oplus \mathbf{e}_{o_2} \\ \cdots \\ \mathbf{e}_{r_T} \oplus \mathbf{e}_{o_T} \end{pmatrix}. \tag{9}$$

Then, we apply multi-head attention module to encode the sequence and finally output the estimated score of the sequence:

$$\begin{aligned} \mathbf{w}_q &= \text{MultiHead}_q(\mathbf{V}_q, \mathbf{V}_q, \mathbf{V}_q), \\ \mathbf{w}_r &= \text{MultiHead}_r(\mathbf{V}_r, \mathbf{V}_r, \mathbf{V}_r), \\ \mathbf{D}_o &= \text{MultiHead}_d(\mathbf{w}_q, \mathbf{w}_r, \mathbf{w}_q \oplus \mathbf{w}_r), \\ score &= \text{Sigmoid}(\text{PredLayer}_d(\mathbf{D}_o)). \end{aligned} \tag{10}$$

The structure of PredLayer_d is identical to PredLayer_g. The final output $score \in [0, 1]$ denotes the probability that the input sequence is a positive sample.

3.4 Adversarial Training

In AdvKT, the generator and the discriminator are alternately updated according to their respective loss functions until convergence. Meanwhile, data augmentation enriches training data to enhance the discriminator's robustness, enabling it to guide the generator in simulating realistic responses through adversarial training.

Generator Loss We train the generator in a multi-step paradigm to minimize the gap between training and inference. The generator has three training goals: (1) aligning predictions with real-world data, (2) making the generated data indistinguishable to the discriminator, and (3) learning more robust and generalized question representations.

The binary cross-entropy (BCE) loss provides a supervised loss to align with the real-world data:

$$\mathcal{L}_{\text{BCE}} = -\sum\nolimits_{t=1}^{T} \left(r_t \log(\hat{r}_t) + (1 - r_t) \log(1 - \hat{r}_t)\right). \tag{11}$$

After generating the responses, the generator obtains a score from the discriminator, and can be adversarially updated by policy gradient [31]:

$$\mathcal{L}_{\text{Adv}} = \frac{1}{T} \sum\nolimits_{t=1}^{T} |\hat{r}_t - 0.5| \times R_t, \tag{12}$$

where $|\hat{r}_t - 0.5|$ is the confidence, and R_t represents the accumulative reward as follows:

$$R_t = -\log\left(1 - \mathcal{D}([(q_i, r_i)]_{i=1}^{t}\right) + \gamma R_{t+1}, \tag{13}$$

where $\gamma \in [0, 1]$ is the decay factor. Moreover, inspired by [18], for more generalized question representations, we also introduce an additional autoregressive loss $\mathcal{L}_{AR}$ to predict the next question q_t based on the student interaction vector $\mathbf{x}_T$:

$$\begin{aligned} \hat{q}_t &= \text{PredLayer}_{gq}(\mathbf{x}_T) \in \mathbb{R}^{|\mathcal{Q}|}, \\ \mathcal{L}_{\text{G-AR}} &= -\sum\nolimits_{t=1}^{T} \sum\nolimits_{i=1}^{|\mathcal{Q}|} \mathbb{I}(q_t = i) \log \hat{q}_t[i]. \end{aligned} \tag{14}$$

Overall, the objective function of the generator is:

$$\mathcal{L}_{\text{G}} = \mathcal{L}_{\text{BCE}} + \lambda_1 * \mathcal{L}_{\text{Adv}} + \lambda_2 * \mathcal{L}_{\text{G-AR}}, \tag{15}$$

where λ_1 and λ_2 are hyperparameters.

Discriminator Loss Due to the data sparsity, we artificially enrich the positive and negative samples based on the student learning pattern to improve the robustness of the discriminator.

Positive samples comprise the original dataset $\mathcal{R}$ and augmented data $\mathcal{T}$. The enrichment of positive samples includes four types: mask, crop, permute, and replace.

- **Mask**: Replace some questions in the origin sequences with a special token [MASK].
- **Crop**: Extract subsequences from the origin sequence.
- **Permute**: Randomly shuffle a subsequence.
- **Replace**: Calculate the difficulty of each question q by

  $$\text{Difficulty}_q = \frac{\#\text{Correctly Answering } q}{\#\text{Answering } q},$$

  and randomly replace questions that the student answers correctly or incorrectly with easier or more difficult ones.

These four augmentation methods generally follow the student learning patterns, ensuring the rationality of the augmented data.

Negative samples include generative data $\mathcal{E}$ and label-reversed data $\mathcal{V}$. The generative data refer to the sequences with responses generated by the generator instead of the ground-truth ones. In addition to the vanilla sequences in the original dataset, we design a heuristic method to generate synthetic sequences to be further labeled by the generator. Specifically, for each question pair (q_A, q_B), we calculate the probability of question q_B occurring after question q_A:

$$p(q_A, q_B) = \frac{\#(q_A, q_B)}{\sum_{q \in \mathcal{Q}} \#(q_A, q)}, \ \forall q_A, q_B \in \mathcal{Q}. \tag{16}$$

We can sample the subsequent questions one by one based on the calculated probabilities above to generate more reasonable synthetic sequences. Furthermore, in addition to the generative data $\mathcal{E}$, we also selectively reverse the binary responses of the sequence in the original dataset $\mathcal{R}$ to create negative samples, denoted $\mathcal{V}$.

The training of the discriminator, based on the enriched positive and negative samples, aims to achieve two objectives: (1) to distinguish between positive and negative samples, and (2) to ensure the stability of the training process. For sequence discrimination, the goal is to maximize the difference between the scores of each positive and negative sample. This is achieved by optimizing the following loss term:

$$\mathcal{L}_{\text{dist}} = -\frac{1}{|\mathcal{R} \cup \mathcal{T}||\mathcal{G} \cup \mathcal{V}|} \sum_{i \in \mathcal{R} \cup \mathcal{T}} \sum_{j \in \mathcal{G} \cup \mathcal{V}} (score_i - score_j). \tag{17}$$

Moreover, to ensure the training stability, we introduce the Gradient Penalty proposed in [11] to restrict the gradient of the discriminator:

$$GP = \alpha \times (\|\nabla_{D_o} \mathcal{D}(q, r)\|_2 - 1)^2, \tag{18}$$

where α is a hyperparameter to balance the gradient penalty. In conclusion, the objective of the discriminator is:

$$\mathcal{L}_{\text{D}} = -\mathcal{L}_{\text{dist}} + GP. \tag{19}$$

Adversarial Learning The general training algorithm is illustrated in Appendix A. We employ an alternating update strategy to iteratively update the discriminator and the generator. When updating the discriminator, we first generate four types of training data with rules, and compute the loss function in Eq. (19), through which the augmented discriminator learns gain robust capabilities. During the training of the generator, it receives rewards from the discriminator, guiding it to simulate more realistic responses, while also being trained by the additional losses in Eq. (15).

Note that the augmented data and label-reversed data are not directly used as the training corpus for the generator, as their sequence properties are highly similar to those in the training set. This similarity limits the effectiveness of GRU-based models. However, it could enhance the attention mechanism by introducing diverse local variations and encouraging more detailed learning of key sequence features, making it more suitable for improving the discriminator.

4 Experiment

4.1 Experimental Setup

Datasets To evaluate the performance of our model, we conduct experiments on four real-world public datasets: ASSIST09, EdNet, Slepemapy and Junyi. Datasets in this paper are from sampling real students' learning logs of different subjects, indicating that the students exhibit different latent learning patterns. The detailed statistics of the datasets are shown in Table 1. The datasets possess varying levels of packing density and distribution, enabling us to test our method's robustness across different data characteristics. We retain students in the dataset who had more than 10 interactions with the platform and select the last 200 records for each student. In order to simulate the scenario with limited data, we randomly select 500 or 6000 students for training, while reserving 20% of the original datasets for testing.

Table 1: Dataset Statistics

Dataset	**ASSIST09**	**EdNet**	**Slepemapy**	**Junyi**
Subject	Math	English	Geography	Math
# Students	500	500	6,000	6,000
# Records	41,741	34,262	553,797	532,139
# Questions	15,003	13,170	4,332	2,164
# Concepts	122	190	1,332	40
Attempts per Q.	2.78	2.60	127.8	245.90
Attempts per C.	342.14	180.32	415.76	13,303.47

Baselines and Evaluation Metrics To evaluate the effectiveness of our model, we compare AdvKT with 15 frequently used KT models. Models trained based on BCE loss include DKT [27], DFKT [24], SAINT [5], EERNNA [30], AKT [8], CKT [29], SAKT [26], GKT [25], DKVMN [35], SKVMN [2], LBKT [33], and simpleKT [20]. Models trained with other objectives include DHKT [32], IEKT [21] and CL4KT [15].

For a fair comparison, these baseline models are also trained under a multi-step setting, *i.e.*, the input of them are question sequences and historically predicted responses. We adopt Accuracy (ACC) and the Area Under Curve (AUC) as metrics. A higher AUC or ACC indicates better performance of the KT task.

Implementation Detail The maximum length of each student's learning sequence is 200. The dimension of the hidden state of GRU is set to 64. The question embedding, concept embedding, response embedding and position embedding all have a dimension of 64. We utilize a 4-headed Transformer in the discriminator. The value of γ in Eq. (12) is chosen in $\{0.9, 0.93, 0.95\ , 0.98\}$. In Eq. (15), λ_1 is set to 1000, and λ_2 is chosen from $\{0, 1\}$. The optimizer used is Adam [**?**]. We update the discriminator every 2 epochs. The learning rate of the generator is 0.001, while the learning rate of the discriminator is 0.005. The memory overhead incurred during training by AdvKT is comparable to that of recent knowledge tracing models, with AdvKT requiring approximately 15,000 MiB, while LBKT [33] utilizes around 12,600 MiB. During inference stage, AdvKT only requires the use of the generator, thereby reducing GPU memory requirements by half. Our model is implemented on PyTorch and is available on Github[6].

4.2 Overall Performance

We compare AdvKT with all baselines under the multi-step training and inference setting. As shown in Table 2, we can obtain the following observations: (1) Baseline models have erratic performance across different datasets. Due to the complexity of student response behaviors, existing models struggle to generalize well when confronted with datasets that emphasize different learning patterns. As a result, each model could only capture specific learning patterns and data characteristics, suffering from multi-step error accumulation and data sparsity. (2) Baselines trained by BCE loss and other objectives show no significant differences compared to those trained with other objectives,

[6] Source code for AdvKT: `https://github.com/fulingyue/AdvKT`

Table 2: Overall performance of AdvKT on four public datasets. The best performing and second-best models are denoted in bold and underlined. * indicates p-value < 0.05 in the significance test.

Groups	Models	ASSIST09		EdNet		Slepemapy		Junyi		Average Performance	
		ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	Avg. ACC	Avg. AUC
BCE Loss	AKT	0.6689	0.6178	0.6318	0.6515	0.7611	0.6520	0.7532	0.7547	0.7037	0.6690
	SAINT	0.6344	0.5272	0.6166	0.6384	0.7605	0.6553	0.7355	0.7599	0.6867	0.6452
	SAKT	0.6127	0.5513	0.6134	0.6333	0.7587	0.6393	0.7319	0.7525	0.6792	0.6441
	CKT	0.6813	0.6342	0.5976	0.5959	0.6884	0.5643	0.7082	0.6930	0.6689	0.6218
	DFKT	0.6426	0.6202	0.6331	0.6589	0.6959	0.5902	0.7291	0.7394	0.6751	0.6522
	DKT	0.6411	0.6188	0.6297	0.6525	0.7474	0.6171	0.7289	0.7360	0.6868	0.6561
	EERNNA	0.6913	0.6182	0.6340	0.6341	0.7535	0.6281	0.7076	0.7161	0.6966	0.6491
	SKVMN	0.6833	0.5515	0.6332	0.6289	0.7428	0.6063	0.7126	0.7334	0.6930	0.6300
	DKVMN	0.6823	0.5323	0.6293	0.6391	0.7609	0.6049	0.7229	0.7265	0.6988	0.6257
	GKT	0.6728	0.5861	0.6001	0.5819	0.6994	0.6039	0.6786	0.6228	0.6627	0.5986
	LBKT	0.6857	0.6249	0.5961	0.6595	0.7603	0.6021	0.6807	0.7108	0.6807	0.6493
	simpleKT	0.6741	0.6402	0.5925	0.6151	0.6976	0.5810	0.7079	0.7318	0.6680	0.6420
Other Objectives	DHKT	0.6898	0.6224	0.6310	0.6303	0.7393	0.6119	0.7167	0.7278	0.6942	0.6481
	IEKT	0.6821	0.6260	0.6376	0.6469	0.6908	0.5853	0.7188	0.7191	0.6823	0.6443
	CL4KT	0.6691	0.6061	0.6151	0.6268	0.7465	0.6295	0.7303	0.7578	0.6902	0.6550
Adv. Learning	AdvKT	**0.6993***	**0.6529***	**0.6464***	**0.6710***	**0.7642***	**0.6724***	**0.7552***	**0.7863***	**0.7163***	**0.6956***

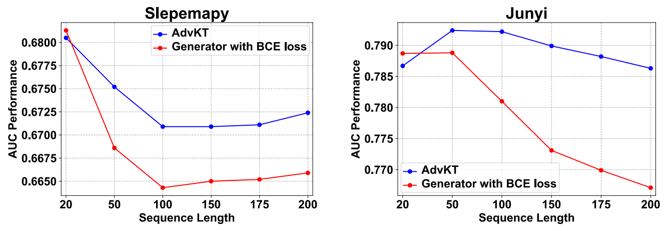


Fig. 3: Comparison results of the influence of learning sequence length of the generator with BCE loss and AdvKT.

because the effectiveness of alternative objectives depends on data distribution and can sometimes introduce noise. (3) AdvKT generally achieves significant performance improvements over the baseline models in both ACC and AUC. On average, our model outperforms the best baseline (AKT) by 1.79% on ACC and 3.98% on AUC. The performance validates the effectiveness of our proposed AdvKT. The adversarial learning paradigm not only help AdvKT acquire students' learning pattern under multi-step prediction scenarios, but also alleviate the data sparsity problem.

4.3 Mitigation of Error Accumulation

In the context of multi-step prediction, error accumulation is a prevalent issue. Typically, in the prediction of long sequences, errors from previous predictions tend to have a more significant impact on the current step. We compare the performance of the generator trained by BCE loss only and our proposed AdvKT under various learning sequence lengths, with AUC as the metric. As depicted in Figure 3, our proposed Ad-

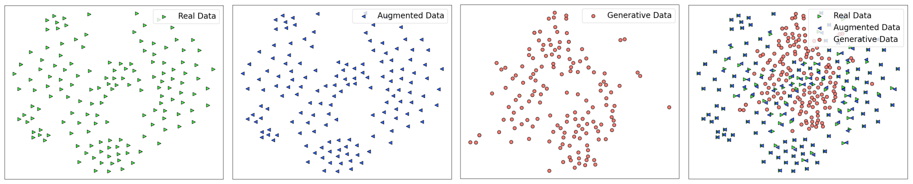


Fig. 4: Distribution of question-response pair visualized with two-dimensional t-SNE on ASSIST09. Green points are real data in dataset. Blue points correspond to data generated by positive data augmentation methods. Red points represent generative data.

vKT enjoys a relatively stable performance with different lengths of sequence, while the generator trained with BCE loss only suffers from severe error accumulation and meets a dramatic performance degeneration as the sequence length gradually increases. The comparison demonstrates the effectiveness of our proposed adversarial learning framework in AdvKT to tackle the error accumulation problem under real-world multi-step prediction scenarios. The generator does not only fit the one-step data distribution via BCE loss, but also expand its horizon upon the entire sequence by receiving sequence-wise reward guidance from the discriminator.

4.4 Mitigation of Data Sparsity

In Figure 4, we visualize the distribution of (q, r) pairs in (i) original data $\mathcal{R}$ in the dataset, (ii) augmented data $\mathcal{T}$ and (iii) generative data $\mathcal{E}$ on four datasets. Our discriminator encodes the entire sequence and obtains hidden state representations. These representation vectors are reduced to two dimensions using t-SNE [22].

The figure shows that the augmented data closely resembles the original data, validating the augmentation methods. However, there are noticeable gaps in both the original and augmented data, and the distribution boundaries are uneven. Although the augmented data increases the data size, it is insufficient to simply combine it with the original data for training a KT model. By generating new sequences, the generative data (red points) fills the gaps and boundary areas of the original data, ensuring a uniform distribution in the problem sequence subspace. Consequently, the discriminator can be trained more stably with the expanded dataset, providing better guidance for the generator to capture comprehensive student learning patterns.

4.5 Ablation Study

Ablation on Data Augmentation To investigate the contribution of data augmentation, we conduct ablation studies on four datasets. We remove the following components to train the discriminator: (i) augmented data $\mathcal{T}$, (ii) label-reversed data $\mathcal{V}$, and (iii) both $\mathcal{T}$ and $\mathcal{V}$ simultaneously. Note that the generative data $\mathcal{G}$ cannot be removed due to the requirement of adversarial learning. The results, shown in Table 3, indicate that the performance of AdvKT decreases regardless of which type of data is removed. We attribute this to the fact that data augmentation provides more diverse data, enhancing

Table 3: Performance comparison between AdvKT and its variants: (i) train discriminator without augmented data $\mathcal{T}$; (ii) train discriminator without reverse label data $\mathcal{V}$; (iii) train discriminator without $\mathcal{T}$ and $\mathcal{V}$. Note that generative data cannot be removed, which is required by adversarial training.

Models	**ASSIST09**		**EdNet**		**Slepemapy**		**Junyi**	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
AdvKT	**0.6993**	**0.6529**	**0.6464**	**0.6710**	**0.7642**	**0.6724**	**0.7552**	**0.7863**
(i) w/o $\mathcal{T}$	0.6895	0.6486	0.6455	0.6693	0.7537	0.6664	0.7518	0.7836
(ii) w/o $\mathcal{V}$	0.6916	0.6501	0.6462	0.6699	0.7564	0.6645	0.7521	0.7843
(iii) w/o $\mathcal{V}$ & $\mathcal{T}$	0.6887	0.6488	0.6460	0.6687	0.7557	0.6610	0.7544	0.7833

Table 4: Ablation studies on different training loss: (i) Remove $\mathcal{L}_{\text{Adv}}$ and train the generator without reward guidance from the discriminator, *i.e.*, supervised learning; (ii) Remove $\mathcal{L}_{\text{BCE}}$ and train the generator only under the supervisions from the discriminator; (iii) Train the discriminator without gradient penalty; (iv) Replace $\mathcal{L}_{\text{dist}}$ for the discriminator with BCE loss.

Models	**ASSIST09**		**EdNet**		**Slepemapy**		**Junyi**	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
AdvKT	**0.6993**	**0.6529**	**0.6464**	**0.6710**	**0.7642**	**0.6724**	**0.7552**	**0.7863**
(i) w/o $\mathcal{L}_{\text{Adv}}$	0.6859	0.6451	0.6462	0.6672	0.7479	0.6659	0.7532	0.7671
(ii) w/o $\mathcal{L}_{\text{BCE}}$	0.5766	0.5299	0.5931	0.5575	0.7083	0.5661	0.7125	0.6588
(iii) w/o GP	0.6735	0.6342	0.6511	0.6683	0.7600	0.6284	0.7541	0.7842
(iv) $\mathcal{L}_{\text{dist}} \to \mathcal{L}_{\text{BCE}}^{\text{dist}}$	0.6908	0.6497	0.6434	0.6698	0.7545	0.6685	0.7463	0.7818

the robustness of the discriminator and offering better reward guidance for the generator to capture student response patterns. Furthermore, augmented data contributes more significantly than label-reversed data, as it generates new sequences of questions.

Ablation on Loss Functions We investigate the impact of different loss functions designed for the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$, with results presented in Table 4.

Firstly, we assess the contribution of loss functions for the generator. Removing either the adversarial loss $\mathcal{L}_{\text{Adv}}$ or the binary cross-entropy loss $\mathcal{L}_{\text{BCE}}$ significantly degrades AdvKT's performance. This suggests the necessity of balancing adversarial learning ($\mathcal{L}_{\text{Adv}}$) with supervised learning ($\mathcal{L}_{\text{BCE}}$) to effectively capture multi-step student response patterns.

Next, we perform an ablation study on the loss functions of the discriminator. We can observe that GP is crucial for maintaining the training stability of the discriminator, ensuring that the generator receives consistent and meaningful reward guidance for enhanced predictive performance. Moreover, replacing our custom-designed distance loss $\mathcal{L}_{\text{dist}}$ with a simple BCE loss $\mathcal{L}_{\text{BCE}}^{\text{dist}}$results in a noticeable performance decline for AdvKT. The distance loss $\mathcal{L}_{\text{dist}}$ focuses more on the internal ranking of positive and negative samples, rather than merely achieving pointwise scores close to 0 or 1 as indicated by the BCE loss. Consequently, our distance loss assigns higher scores to positive samples than negative ones, ensuring more accurate reward guidance for the generator.

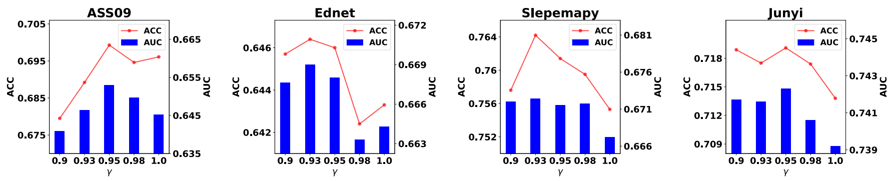


Fig. 5: Ablation study on the hyper-parameter γ on four datasets.

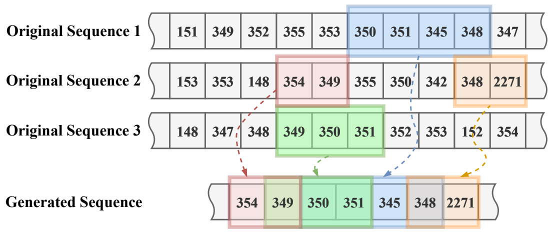


Fig. 6: Demonstration of generated question sequences and original question sequences.

Ablation on Discount Factor γ To assess the sensitivity of our model, we evaluate the impact of discount factor γ. We test our method with $\gamma = \{0.9, 0.93, 0.95, 0.98, 1\}$ across four datasets, keeping other hyperparameters constant. As illustrated in Figure 5, the performance fluctuates when tuning the discount factor. The discount factor γ determines the importance the generator places on future predictive performance and should be adjusted based on the specific dataset requirements.

4.6 Insight of Adversarial Learning

Generative data $\mathcal{G}$ plays a crucial role in training the generator alongside the real dataset. The quality and diversity of these synthetic sequences directly influence the supervision signal ($\mathcal{L}_{\text{adv}}$) obtained by the generator. In Figure 6, we present a generated synthetic sequence along with its most similar learning sequences from the dataset. Four subsequences are found within the original sequences, highlighting a certain similarity between the generated and original sequences. However, each generated sequence remains distinct from any single original sequence.

This similarity ensures that the generated sequences maintain the logical consistency of the original data, such as presenting simpler questions before more challenging ones. Meanwhile, the diversity in the generated sequences helps AdvKT tackle data sparsity by enriching the dataset with varied examples. This diversity also enables the model to learn from accumulated errors across different sequences, further reducing the impact of error accumulation.

5 Related Work

5.1 Knowledge Tracing

To investigate students' learning patterns, numerous knowledge tracing models have been proposed under a single-step training paradigm.

Deep Knowledge Tracing (DKT) [27] pioneers using the deep learning-based knowledge tracing methods. It uses the hidden states of long short-term memory (LSTM) [14] networks to describe the students' knowledge states. After DKT, DHKT [32] and other variants of DKT introduce extra information like question-skill relations and additional regularization terms to improve the performance. Inspired by Key-Value Memory Network (KVMN) [23], several works [35,2] use key-matrix and value-matrix to represent concepts and student's mastery level. GKT randomly builds a similarity graph of skills, GIKT [34] applies the graph of skills and questions to obtain better representations of questions. After the transformer architecture was proposed, some works [5,30] tried to utilize the attention mechanism in KT tasks.

Recently, StableKT [17] highlighted the presence of error accumulation in single-step KT scenarios. All the aforementioned methods experience more severe error accumulation under multi-step inference. At the same time, they suffer significant performance degradation when training data is sparse. Although approaches [12,7,4] enhance learning from question text using large language models, educational datasets still face challenges at the ID level, with limited and uneven interaction data. AdvKT utilizes an adversarial learning paradigm, thereby improving both training paradigms and data utilization.

5.2 Adversarial Learning

In recent years, GAN [10] has been widely used for data augmentation in computer vision and Natural Language Processing. Combining the idea of GAN with Inverse Reinforcement Learning (IRL), researchers propose GAIL [13] framework to learn policy by imitating expert trajectories. Many previous work [3,28,6] adopt GAN and GAIL frameworks for user modeling. These work models and simulates users' behavior (clicking, buying) on the web search page or shopping websites.

Unlike user interest modeling, the KT task includes the change of users' state, *i.e.*, knowledge state transition. In AdvKT, we consider the students' learning patterns and generate logical question sequences to estimate the knowledge state of students.

6 Conclusion

In this work, we propose a novel adversarial multi-step training framework for knowledge tracing (AdvKT), which, for the first time, explicitly models real-world multi-step prediction by introducing the adversarial learning framework. AdvKT adversarially trains a generator and a discriminator, aiming at alleviating error accumulation and data sparsity problem. The generator is designed to simulate students' learning process under the multi-step setting, *i.e.*, generating students' responses based on the question sequence and its previous predicted responses instead of the ground-truth ones. The discriminator, whose training data is augmented by well-designed rules, distinguishes whether the learning sequence is derived from real data or fake data, and therefore provides sequence-wise reward guidance for the generator to capture student response patterns. Extensive experiments on four real-world datasets demonstrate the superiority of AdvKT compared with existing baseline models, as well as its capabilities for mitigating the error accumulation and data sparsity problem.

Acknowledgments. The Shanghai Jiao Tong University team is partially supported by National Key R&D Program of China (2022ZD0114804), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102) and National Natural Science Foundation of China (62322603, 62177033).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
2. Abdelrahman, G., Wang, Q.: Knowledge tracing with sequential key-value memory networks (2019)
3. Bai, X., Guan, J., Wang, H.: Model-based reinforcement learning with adversarial training for online recommendation (2019)
4. Cao, Y., Zhang, W.: Mamba4kt:an efficient and effective mamba-based knowledge tracing model (2024)
5. Choi, Y., Lee, Y., Cho, J., Baek, J., Kim, B., Cha, Y., Shin, D., Bae, C., Heo, J.: Towards an appropriate query, key, and value computation for knowledge tracing (2020)
6. Dai, X., Lin, J., Zhang, W., Li, S., Liu, W., Tang, R., He, X., Hao, J., Wang, J., Yu, Y.: An adversarial imitation click model for information retrieval (2021)
7. Fu, L., Guan, H., Du, K., Lin, J., Xia, W., Zhang, W., Tang, R., Wang, Y., Yu, Y.: Sinkt: A structure-aware inductive knowledge tracing model with large language model. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. p. 632–642. CIKM '24, ACM (Oct 2024)
8. Ghosh, A., Heffernan, N., Lan, A.S.: Context-aware attentive knowledge tracing. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining. p. 2330–2339. KDD '20, Association for Computing Machinery, New York, NY, USA (2020)
9. Gong, J., Wang, S., Wang, J., Feng, W., Peng, H., Tang, J., Yu, P.S.: Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. pp. 79–88 (2020)
10. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks (2014)
11. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 5769–5779. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)

12. Guo, Y., Shen, S., Liu, Q., Huang, Z., Zhu, L., Su, Y., Chen, E.: Mitigating cold-start problems in knowledge tracing with large language models: An attribute-aware approach. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. pp. 727–736 (2024)
13. Ho, J., Ermon, S.: Generative adversarial imitation learning (2016)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997)
15. Lee, W., Chun, J., Lee, Y., Park, K., Park, S.: Contrastive learning for knowledge tracing. In: Proceedings of the ACM Web Conference 2022. p. 2330–2338. WWW '22, Association for Computing Machinery, New York, NY, USA (2022)
16. Li, Q., Xia, W., Yin, L., Shen, J., Rui, R., Zhang, W., Chen, X., Tang, R., Yu, Y.: Graph enhanced hierarchical reinforcement learning for goal-oriented learning path recommendation. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. pp. 1318–1327 (2023)
17. Li, X., Bai, Y., Guo, T., Liu, Z., Huang, Y., Zhao, X., Xia, F., Luo, W., Weng, J.: Enhancing length generalization for attention based knowledge tracing models with linear biases. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI). pp. 5918–5926. International Joint Conferences on Artificial Intelligence (2024)
18. Liotet, P., Venneri, E., Restelli, M.: Learning a belief representation for delayed reinforcement learning. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021)
19. Liu, Q., Tong, S., Liu, C., Zhao, H., Chen, E., Ma, H., Wang, S.: Exploiting cognitive structure for adaptive learning. pp. 627–635 (07 2019)
20. Liu, Z., Liu, Q., Chen, J., Huang, S., Luo, W.: simplekt: a simple but tough-to-beat baseline for knowledge tracing (2023)
21. Long, T., Liu, Y., Shen, J., Zhang, W., Yu, Y.: Tracing knowledge state with individual cognition and acquisition estimation. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 173–182. SIGIR '21, Association for Computing Machinery, New York, NY, USA (2021)
22. Van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(11) (2008)
23. Miller, A., Fisch, A., Dodge, J., Karimi, A.H., Bordes, A., Weston, J.: Key-value memory networks for directly reading documents (2016)
24. Nagatani, K., Zhang, Q., Sato, M., Chen, Y.Y., Chen, F., Ohkuma, T.: Augmenting knowledge tracing by considering forgetting behavior. pp. 3101–3107 (05 2019)
25. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In: IEEE/WIC/ACM International Conference on Web Intelligence. p. 156–163. WI '19, Association for Computing Machinery, New York, NY, USA (2019)
26. Pandey, S., Karypis, G.: A self-attentive model for knowledge tracing (2019)
27. Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., Sohl-Dickstein, J.: Deep knowledge tracing (2015)
28. Shi, J.C., Yu, Y., Da, Q., Chen, S.Y., Zeng, A.X.: Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning (2018)
29. Shuanghong, S., Liu, Q., Chen, E., Wu, H., Huang, Z., Zhao, W., Su, Y., Ma, H., Wang, S.: Convolutional knowledge tracing: Modeling individualization in student learning process. pp. 1857–1860 (07 2020)
30. Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., Ding, C., Wei, S., Hu, G.: Exercise-enhanced sequential modeling for student performance prediction. In: Proceedings of the

Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI'18/IAAI'18/EAAI'18, AAAI Press (2018)
31. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems **12**, 1057–1063 (2000)
32. Wang, T., Ma, F., Gao, J.: Deep hierarchical knowledge tracing. In: Lynch, C., Merceron, A., Desmarais, M., Nkambou, R. (eds.) EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining. pp. 671–674. EDM 2019 - Proceedings of the 12th International Conference on Educational Data Mining, International Educational Data Mining Society (2019)
33. Xu, B., Huang, Z., Liu, J., Shen, S., Liu, Q., Chen, E., Wu, J., Wang, S.: Learning behavior-oriented knowledge tracing. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. p. 2789–2800. KDD '23, Association for Computing Machinery, New York, NY, USA (2023)
34. Yang, Y., Shen, J., Qu, Y., Liu, Y., Wang, K., Zhu, Y., Zhang, W., Yu, Y.: Gikt: A graph-based interaction model for knowledge tracing (2020)
35. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing (2016)