



SATCN: An Improved Temporal Convolutional Neural Network with Self Attention Mechanism for Knowledge Tracing

Ruixin Ma^{1,2}, Hongyan Zhang^{1,2}, Biao Mei^{1,2}, Guangyue Lv^{1,2},
and Liang Zhao^{1,2}(✉)

¹ Dalian University of Technology, Dalian 116600, China
liangzhao@dlut.edu.cn

² The Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian, China

Abstract. With the rapid expansion of E-education, knowledge tracing (KT) has become a fundamental mission which traces the formation of learners' knowledge states and predicts their performance in future learning activities. Knowledge states of each learner are simulated by estimating their behavior in historical learning activities. There are often numerous questions in online education systems while researches in the past fails to involve massive data together with negative historical data problems, which is mainly limited by data sparsity issues and models. From the model perspective, previous models can hardly capture the long-term dependency of learner historical exercises, and model the individual learning behavior in a consistent manner is also hard to accomplish. Therefore, in this paper, we develop an Improved Temporal Convolutional Neural Network with Self Attention Mechanism for Knowledge Tracing (SATCN). It can take the historical exercises of each learner as input and model the individual learning in a consistent manner that means it can realize personalized knowledge tracking prediction without extra manipulations. Moreover, with the self attention mechanism our model can adjust weights adaptively, thus to intelligently weaken the influence of those negative historical data, and highlight those historical data that have greater impact on the prediction results. We also take attempt count and answer time two features into account, considering proficiency and forgetting of the learners to enrich the input features. Empirical experiments on three widely used real-world public datasets clearly demonstrate that our framework outperforms the presented state-of-the-art models.

Keywords: Knowledge Tracing · Temporal Convolutional Neural Network · Self Attention Mechanism

1 Introduction

Knowledge Tracing(KT) is a foremost task of on-line education, which propose to trace learners' knowledge states based on their historical learning trajectories. The success of knowledge tracing can benefit both personalized learning

and adaptive learning so that has attracted prodigious attention over the past decades [1–3].

The KT task can be defined as a supervised sequential sequence learning task: according to learners’ historical exercise interactions $X = \{x_1, x_2, \dots, x_t\}$, predicting their future interaction x_{t+1} [4]. In the question-answering system, the t^{th} interaction is expressed as a tuple $x_t = (q_t, a_t)$, where q_t is the label of exercise that the learner attempts at a certain timestamp t and a_t is the correctness of the learner’s answer about q_t . $a_t \in \{0, 1\}$, where correct answer is recorded as 1 and incorrect answer is recorded as 0. The purpose is predicting the probability of learners will be able to answer the future exercise correctly, i.e., predicting $p(a_{t+1}|q_{t+1}, X_t)$.

In the previous studies, many efforts have been made towards knowledge tracing. Among them, Bayesian Knowledge Tracing (BKT) is one of the most representative early works [2]. However, BKT is a highly restricted structured model. Recent years, benefit from the high capacity and effective representation learning of deep neural networks, the first KT model based on deep learning named Deep Knowledge Tracing (DKT) [4] has become BKT’s alternative model, and its excellent performance boost leads us to inspect limitations of BKT. DKT is the first KT method based on deep learning that utilizes Long Short-term Memory (LSTM) [5] which is an excellent variant of recurrent neural networks (RNNs) to trace learners’ knowledge states. The latest progress in deep learning has developed a series of deep KT models. Two representative deep KT models are Dynamic Key-Value Memory Networks (DKVMN) [6] and Sequential Key-Value Memory Networks (SKVMN) [7], which leverages Memory-Augmented Neural Networks (MANNs) [8] and hop-LSTM respectively to solve knowledge tracing.

For the purpose of learning high-quality KT models, a substantial amount of comprehensive data is inevitably required for ensuring the stability of neural networks during training. However, the practical educational scenes usually encounter negative historical data problem, which is ubiquitous in learning process. To be more specific, as shown in Fig. 1, a learner want to solve an exercise on “Quadratic equation” (e_5) which belongs to the knowledge concept “Equations”. We can see that there are four exercises have been finished, and they are linear equations (e_1), arithmetic (e_2), plane geometry (e_3), and solid geometry (e_4) respectively. The goal of prediction is whether quadratic equation(e_5) can be done right. Obviously, the correctness of the first two exercises should have greater impact on the prediction result than the last two exercises. However, the impact of each exercise is often treated equally, which is apparently irrational. In addition, if the correctness of the latest two exercises (e_3, e_4) is not optimistic, then they would become negative historical data for prediction results.

Therefore, in this paper, we conduct principled studies on above issues, and propose an Improved Temporal Convolution Neural Network with Self-Attention Mechanism for Knowledge Tracing (SATCN). Firstly, SATCN can accomplish the time series prediction task while capturing long term dependency, which can take full advantage of time information in learning data. In addition, the input

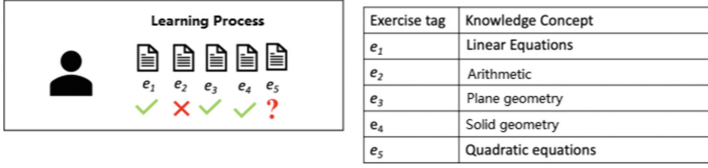


Fig. 1. Left sunfigure shows sequence of exercises that one learner attempted in a timestep and right sunfigure demonstrates knowledge concept corresponding to each exercise.

sequence is each learner’s exercises they finished instead of treating all learners as a whole or taking much fewer skills as input like previous models, which eases the problem of data sparsity to a certain extent. This is because lots of questions only correspond to much fewer skills. This is also why our model can achieve personalization without additional processing by training and testing each learner’s exercise sequence in a consistent manner. We also consider the attempt count and answer time to simulate learners’ proficiency and forgetting to enrich the input features. It can be observed from the original data that learners often give correct answer in the first attempt if they have already mastered related knowledge, while questions they tried many times are rarely answered correctly. Thus, we can speculate that learners haven’t mastered the relevant knowledge but want to guess the correct answer. Answer time is the time interval between learners start answering questions and submitting answer. It reflects learners’ proficiency in the relevant knowledge to a certain extent. As we all known, the more proficient in the relevant knowledge, the more likely they answer the questions correctly and harder to forget. We build a self-attention layer between the input layer and the hidden layer to enable the model can adjust weights adaptively, then it is able to deal with a series of problems caused by negative historical data, so as to achieve more accurate predictions.

Our main contributions are summarized as follows:

- (1) The KT task is creatively described as a time series forecasting task and an Improved Temporal Convolutional Neural Network with Self Attention Mechanism for Knowledge Tracing (SATCN) is proposed by improving DeepTCN architecture, which can better simulate the learning process of learners.
- (2) By designing a novel data processing and training method, our model can relieve the problem of data sparseness and can model learning process in a consistent manner. At the same time, it can achieve personalized prediction without requiring other operations.
- (3) Our model can automatically discover and reduce the influence of negative data on prediction results by accomplished with Self Attention Mechanism. Simultaneously, by considering the characteristics of data we take proficiency and forgetting into consideration to enrich the input features and bring a performance boost of our model.

2 Related Work

2.1 Knowledge Tracing

KT is a necessary task in Intelligent Education System (ITS) [9]. The current knowledge tracing methods can be roughly divided into three categories: the first is KT methods based on probability graph, then KT methods based on logistic regression and KT methods based on deep learning. We mainly study the last KT methods in this paper.

KT methods based on probability graph mainly refer to Bayesian Knowledge Tracing (BKT) [2]. BKT is a Hidden Markov Model (HMM) with hidden variables. Subsequential works include contextualization of slip and guess probability estimation [10], problem difficulty [11], and personalization [12, 13]. The most commonly used KT methods based on logistic regression are Item Response Theory (IRT) [14] and Performance Factors Analysis (PFA) [15]. The main idea of IRT is to estimate the probability that learners can answer questions correctly base on their learning ability and item’s difficulty. While PFA expands the static knowledge assumption, and it can model multiple features simultaneously with its basic structure.

In 2015, Chris Piech et.al. firstly applied RNNs in KT models which increased the AUC by 25%. The appearance of DKT [4] subverts BKT’s transcendent status in KT area. Since then, KT models based on deep learning have attracted the attention of large number researchers. Apart from enriching model features, subsequent improvement works further rely on more powerful neural networks. For example, DKVMN [6] has one static matrix called key and the other dynamic matrix called value, which unlike standard memory-augmented neural networks only supports a single memory matrix or two static memory matrices. Therefore, DKVMN can solve the problem that existing methods fail to accurately determine which concepts learners are good at or unfamiliar with when modeling the knowledge state of each predefined concept. SAKT [16] is the first model uses the attention mechanism, which can solve the problem that other models can’t generalize well when dealing with sparse data. SAINT [17] imitates the encoder-decoder structure of Transformer [18], and inputs exercise and response separately, where the exercise sequence is fed into encoder, the outputs of encoder and response sequence are fed into decoder. Several works [19, 20] try to introduce graph structure into knowledge tracking, so that KT models can process non-Euclidean data. GKT [19] uses graph neural network [21] to transform the KT task into a time-series node-level classification mission in graph structure. While GIKT [20] utilizes three-layer Graph Convolutional Network (GCN) [22] to generalize the high-order relationship between problems and skills before training.

However, all of the above models face the same data sparsity problem with few features, and don’t make full use of the temporal information in the data. Negative historical data problems are ignored in these models, and they can’t model the individual learning behavior in a consistent manner. In this paper, our model alleviates the data sparseness from two perspectives: data processing and

feature enrichment, which enable our model simulates proficiency and forgetting while simulating interactions. The network architecture that can make full use of implicit time information for training would be used, at the same time, it can adjust weights adaptively to solve negative historical data problems with self-attention mechanism.

2.2 Deep Temporal Convolutional Neural Network

DeepTCN [23] is a probabilistic forecasting framework based on convolutional neural network (CNN) [24] for multiple related time series forecasting. This framework consists of stacked residual blocks based on dilated causal convolutional nets with encoder-decoder structure, which are constructed to capture the temporal dependencies of input series with expansive receptive field and to capture long-range temporal dependencies with fewer number of layers. DeepTCN has powerful representation learning capabilities, so that it is capable of learning complex patterns like regularity, internal and inter-date influence, and to exploit those patterns for more accurate forecasts. When exercises data is sparse or unavailable this capability will show its powerful ability which is common in our study.

A general probabilistic forecasting problem for multiple related time series can be described as follows: Given a set of time series $y_{1:t} = \{y_{1:t}^{(i)}\}_{i=1}^N$, the future time series are $y_{(t+1):(t+\Omega)} = \{y_{(t+1):(t+\Omega)}^{(i)}\}_{i=1}^N$, where N is the number of series, t is the length of the historical observations and Ω is the length of the forecasting horizon. Therefore, prediction results and be described as:

$$P(y_{(t+1):(t+\Omega)}|y_{1:t}) = \prod_{\omega=1}^{\Omega} p(y_{t+\omega}|y_{1:t}, X_{t+\omega}^{(i)}, i = 1, \dots, N) \quad (1)$$

Our primary motivation for using DeepTCN is the success of CNNs and adaptability of DeepTCN in this task. DeepTCN is a non-autoregressive probability prediction framework for great quantity correlated time series, which can learn the latent correlations among sequences and model their interactions. Specially, it is able to handle data sparsity and cold start problems that are common in complex real-world forecasting situations, and has high scalability and extensibility. Extensive empirical researches show that DeepTCN outperforms the state-of-the-art methods in both point prediction and probability prediction tasks. By using this framework, our model can capture long-term dependencies with expansive receptive field, and it can excavate temporal information in data automatically. Especially, it's able to deal with historical data sparse or unavailable problem well which is ubiquitous in KT task.

3 Proposed Method

3.1 Input Representation

Our model is used to predict whether learners will be able to answer the future exercise x_{t+1} correctly based on their historical interactions $X = \{x_1, x_2, \dots, x_t\}$.

The architecture of SATCN is shown in Fig. 2, the interaction tuple x_t is a quintuple tuple, and the meaning of each notation is presented in Table 1.

Table 1. Notations

Notations	Description
s_t	the t^{th} learner in all learners
e_t	the t^{th} exercise solved by the learner
a_t	the correctness of e_t
ac_t	the attempt count of e_t
at_t	the amount of time that a learner spent on e_t

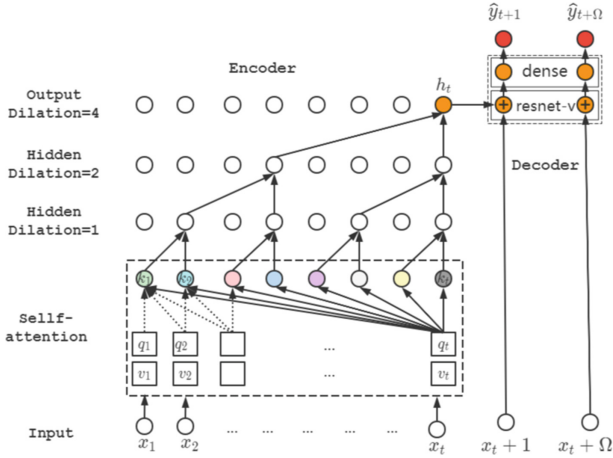


Fig. 2. The architecture of SATCN.

The embedding layer in SATCN maps each input vector x_i to latent space to generate its embedding vector. Specifically, it maps all the five features of x_i to high-dimensional vector space, then the connection operation is performed, which aims to turn the vector x_i into an embedding vector with a fixed dimension of 512. At each timestamp $t + 1$, embedding layer uses Exercise embedding to embed the exercise that the learner is currently trying to solve into the query space to obtain the corresponding query q_i , and uses Interaction embedding to embed historical interactions x_t into the key and value space to obtain the corresponding k_i and v_i (see Fig. 3).

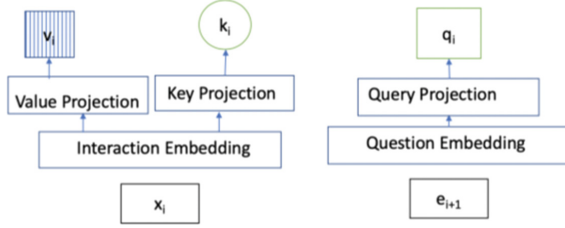


Fig. 3. Embedding layer.

3.2 Self-attention Layer

The purpose of self-attention layer is to acquire the query, key, and value corresponding to the input from the embedding layer, and calculate the attention weight. We use Q_{in}, K_{in} and V_{in} represent the sequence of query, key and value respectively. The multi-head attention network utilizes different projection matrices to perform h different projections on the same input sequence to maps Q_{in}, K_{in} and V_{in} into the latent space, which can be described as:

$$Q_i = [q_1^i, \dots, q_k^i] = Q_{in} W_i^Q \quad (2)$$

$$K_i = [k_1^i, \dots, k_k^i] = K_{in} W_i^K \quad (3)$$

$$V_i = [v_1^i, \dots, v_k^i] = V_{in} W_i^V \quad (4)$$

Herein, q, k, v represent projected queries, keys and values respectively. We employ scaled dot-product attention mechanism. The correlation between value and given query is determined by the dot product between query and the key corresponding to the value.

There are total 8 attention heads and each of them ($head_i$) is the product of matrix $\frac{Q_i K_i^t}{\sqrt{d}}$ after Softmax operation and values V_i .

$$head_i = Softmax\left(\frac{Q_i K_i^t}{\sqrt{d}}\right) V_i \quad (5)$$

where d is the dimension of query and key, which is used to scale.

The 8 attention heads multiply by W^O after connection to aggregate the output of different attention heads, which is the final output of multi-head attention network, i.e., attention weights.

$$MultiHead(Q_{in}, K_{in}, V_{in}) = Concat(head_1, \dots, head_h) W^O \quad (6)$$

3.3 Encoder: Dilated Causal Convolutions

In the encoder part, stacked dilated causal convolutions are constructed to simulate the stochastic process of historical observation and output $h_t^{(i)}$. Causal convolutions refer to the convolutions that the output at time can only be obtained

from the input, similar to the Mask mechanism in Transformer. Dilated causal convolutions allow the application of filters in a range of more than its length by skipping input value with a certain step. In the situation of univariate series, given the one-dimension input series x , the output of dilated convolutions at location t with kernel ω is feature map s , which can be described as:

$$s(t) = (x *_d \omega)(t) = \sum_{k=0}^{K-1} \omega(k)x(t - d \cdot k) \quad (7)$$

where d is the dilation factor, and K is the kernel size. The network can retain very broad receptive field by stacking more than one dilated convolution, and capture long-term temporal dependencies with less layers. As shown in Fig. 2, there are dilated causal convolutions with dilation factors $d = \{1, 2, 3\}$ in the left, and the size of the kernel $K = 2$. The receptive field with a size of 8 is constructed by stacking three layers.

The details of each layer in encoder are shown in Fig. 4, where there are two dilated convolution modules in each layer, and they are exactly the same (with same kernel size K and dilation factor d). Every two layers of dilated causal convolutions assemble a residual block. Each layer of dilated causal convolutions is followed by batch normalization and rectified nonlinear unit (ReLU). The difference is that the output of the batch normalization which at the location of the second dilated causal convolution layer would be used as the input of the residual block, then the second ReLU. In the case of capturing long-term dependencies, the residual block helps train the network more stable and efficient. At the same time, the rectified nonlinear unit (ReLU) is corrected to obtain better prediction accuracy.

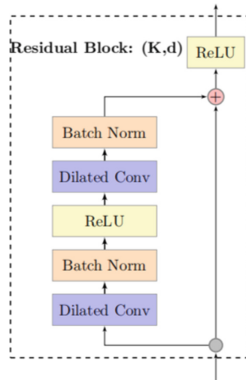


Fig. 4. Encoder module.

3.4 Decoder: Residual Neural Network

The decoder module is composed of Resnet-v and a dense layer. The Resnet-v module is the variant of residual neural network, which is used to capture the information of two inputs (one is hidden outputs and the other is exogenous variables) from the encoder. It can be described as:

$$\delta_{t+\omega}^{(i)} = R(X_{t+\omega}^{(i)}) + h_t^{(i)} \quad (8)$$

In which $h_t^{(i)}$ is hidden outputs, $X_{t+\omega}^{(i)}$ is exogenous variables, and the hidden outputs of resnet-v is $\delta_{t+\omega}^{(i)}$. $R(\bullet)$ is the residual function acting on exogenous variables. It explains the residuals between the predict value and the true value, and plays the role of transfer function at the same time. Simply speaking, resnet-v combines hidden outputs $h_t^{(i)}$ and exogenous variables $X_{t+\omega}^{(i)}$ processed by residual function, and generates new hidden outputs.

Figure 5 shows the detailed structure of the decoder. The first dense layer and batch normalization are used to project the exogenous variables $X_{t+\omega}^{(i)}$, then ReLU is used for activation. There are the second dense layer and batch normalization subsequently. All of them construct the residual function $R(X_{t+\omega}^{(i)})$ acting on exogenous variables $X_{t+\omega}^{(i)}$. At last, use a dense layer to map the hidden outputs $\delta_{t+\omega}^{(i)}$ that generated by Resnet-v to obtain the predicted probability.

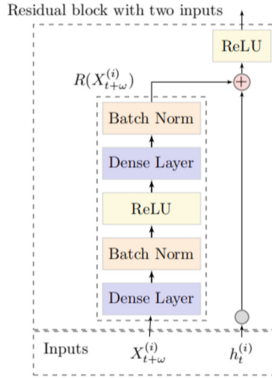


Fig. 5. Decoder module.

3.5 Network Training

The model is trained by minimizing the quantile loss $L_q(y, \hat{y}^q)$. For a specific quantile q , the true value and predicted value is y and \hat{y}^q respectively, and its loss function can be described as:

$$L_q(y, \hat{y}^q) = q(y - \hat{y}^q)^+ + (1 - q)(\hat{y}^q - y)^+ \quad (9)$$

where $(y)^+ = \max(0, y)$ and $q \in (0, 1)$. Given a set of quantile levels $Q = \{q_1, \dots, q_m\}$, the objective of training is to minimize the total quantile loss L_Q :

$$L_Q = \sum_{j=1}^m L_{q_j}(y, \hat{y}^{q_j}) \quad (10)$$

4 Experiments

4.1 Datasets

ASSIST2009: This dataset is collected from ASSISTments in 2009 ~ 2010 school year which is an online education platform. We screen learners on the condition of each learner’s exercise sequence no less than 400, and then intercept the first 400 as the historical exercise sequence of each learner.

ASSIST2012: This dataset is also collected from ASSISTments, which contains learning data in 2012 ~ 2013 school year. We select exercise sequence of each learner in case of sequence length no less than 400.

Junyi Academy: This dataset is collected from Junyi Academy in 2015, which is an online education website provides learning materials and practice platform about various science subjects. We totally choose 800 learners and their exercise sequence length no less than 800.

The statistical data of all datasets are shown in Table 2.

Table 2. The statistical data of all datasets

	ASSIST2009	ASSIST2012	Junyi Academy
Student Number	876	800	800
Question Number	24879	31689	616
Exercise Number	400	400	800
Attempt Count	400	29	333
Answer Time(ms)	173301	11864	262582

4.2 Baseline Models

The Baseline Models we compared are:

DKT [4]: Deep Knowledge Tracing (DKT) is the first KT model based on deep learning, which utilizes LSTM of recurrent neural network to process KT task.

DKT+ [25]: This model is the variant of DKT, which introduce regularization terms corresponding to reconstruction and waviness into the loss function of DKT to robust the consistency of prediction.

DKVMN [6]: This model improves MANN by using Dynamic Key-Value Memory to construct Dynamic Key-Value Memory Network, which is been recognized as an excellent KT improvement model based on deep learning.

SAKT [16]: Inspired by Transformer, it is the first KT model use self-attention mechanism.

4.3 Implementation Details

We take the correctness of each question as the main element, and the remaining four data (learner id, question id, attempt count and answer time) are embed as covariates. Suppose the exercise sequence length of each learner is Q , then the input data is a $Q * 5$ matrix, and each column represents learner id, question id, correctness, attempt count and answer time respectively. The input data will become a vector with 512 dimensions after embedding layer, and we train and test for every single learner. We uniformly set the last 30 exercise sequence as test datasets, and the other $Q - 30$ exercise sequence as train datasets.

We set LR as 0.001, and keep the dropout rate as 0.1 to avoid overfitting. The performance of all models evaluate by calculating the area under the ROC curve (AUC) and its standard deviation. We randomly choose 10 consecutive epochs from the result after the model converged and report the mean and standard deviation of AUC.

4.4 Results and Analysis

The area under the ROC curve(AUC) demonstrate every model’s prediction performance: AUC with high value means high prediction accuracy and better prediction performance, and the standard deviation of AUC reflects model’s stability: the smaller the standard deviation, the better the stability. We compare our model with standard DKT, DKT+, DKVMN and SAKT model, and the overall performance of all models is shown in Table 3.

Table 3. Test AUC for all datasets

Datasets	ASSIST2009		ASSIST2012		Junyi Academy	
Model	mean	std	mean	std	mean	std
DKT [4]	0.797	0.0099	0.669	0.0082	0.855	0.0246
DKT+ [25]	0.787	0.0089	\	\	\	\
DKVMN [6]	0.672	0.0214	0.682	0.0014	0.574	0.0524
SAKT [16]	0.704	0.0117	0.718	0.0106	0.795	0.0611
SATCN[our]	0.811	0.0005	0.800	0.0011	0.874	0.0003

In ASSIST09, the average AUC of DKT can reach 0.797, but its variant model DKT+ only reaches 0.787. DKVMN and SAKT reach 0.672, 0.704 respectively,

and SATCN reaches the best average AUC with 0.811. Comparing with DKT, DKT+, DKVMN and SAKT, SATCN improves by 1.76%, 3.05%, 20.68% and 15.20%. At the same time, DKVMN shows the most obvious fluctuation with the maximum standard deviation. The other 3 baseline models' stability is better but still inferior to SATCN. Our SATCN model surpasses previous models both in accuracy and stability at this data set.

In ASSIST12, the performance of DKT is severely degraded and its AUC only reaches 0.669, while the performance and stability of DKVMN and SAKT have lightly improved, and their AUC reach 0.682 and 0.718 respectively. Our model reaches 0.800 that improve by 19.58%, 17.30% and 14.42% comparing with DKT, DKVMN and SAKT. Meanwhile the stability of SATCN reaches the best performance.

In JunyiAcademy, DKT and SAKT represent good AUC with 0.855 and 0.795, which means they show the best model performance. But DKVMN displays poor AUC with 0.574, and all baseline models exhibit the worst stability. While our model shows the best AUC with 0.874, and the best stability simultaneously. Comparing with DKT, DKVMN and SAKT, SATCN improves by 2.22%, 52.26% and 9.94%, respectively.

Restricted by devices, memory space and computing power DKT+ model need in ASSIT12 and JunyiAcademy can't be satisfied, so we couldn't get the result.

We can summarize from the experimental results in Table 3 that DKT, SAKT and SATCN reach the best performance in JunyiAcademy which contains the longest interaction sequence and the minimal question type, and they show poorer performance in ASSIST09 and ASSIST12 which include shorter interaction sequence and more question type. We find that the more question type involved under the same length of interaction sequence length, the worse the performance, while DKVMN shows the opposite effect.

We think longer interaction sequence and less question type means more frequent interactions in each question, and neural network is good at capturing these interactions. Meanwhile the more question type involved under the same length of interaction sequence length means many questions would be answer only once and their interaction is not obvious. Therefore, more details about the above experimental results are revealed in Table 3. The original work of DKVMN [6] also find that it will achieve better performance when there are large number of different questions. We think it's involved with external memory, because it can capture low-frequent interactions better than neural network. Even if the neural network captures low-frequent interactions, it also has great probability of forgetting and dropout.

To conclude, the performance of SATCN in all datasets outperform other baseline models, especially in JunyiAcademy which contains more frequent interactions about each exercise. The experimental results indicate that our model can capture exercise interactions better and make more accurate prediction.

4.5 Ablation Studies

To explore the influence of answer time and attempt count about experimental results, we conduct ablation studies at the four variants of SATCN, SATCN-AC, SATCN-AT, SATCN-BA, and their detail settings are shown as bellow. Their performance is shown in Table 4. For SATCN-AC, we remove attempt count related to each exercise. For SATCN-AT, we remove answer time related to each exercise. For SATCN-BA, we remove both attempt count and answer time related to each exercise.

Table 4. Effect of Ablation Models

Datasets	ASSIST2009		ASSIST2012		Junyi Academy	
Models	mean	std	mean	std	mean	std
SATCN-BA	0.765	0.0007	0.740	0.0041	0.840	0.0007
SATCN-AC	0.758	0.0017	0.745	0.0014	0.851	0.0005
SATCN-AT	0.839	0.0007	0.797	0.0012	0.853	0.0030
SATCN	0.811	0.0005	0.800	0.0011	0.874	0.0003

From above results, we can see that SATCN-BA shows the most significant performance degradation, which remove both attempt count and answer time. Both attempt count and answer time show a certain positive effect on the improvement of performance. We can also seen that compared with answer time, attempt count has greater impact on experimental results. This result proves our conjecture: answer time can model learner’s proficiency, combined with attempt count our model can accurately simulate the knowledge state of a learner come to understand whether they has mastered or not and their proficiency. This level of proficiency is also can be used to simulate forgetting, thereby improving the performance of our model.

5 Conclusions and Future Work

In this paper, we propose a new learning framework named SATCN from the perspective of temporal sequence prediction to deal with KT task. Our model can be used in actual education scenarios to help tutors and learners to improve teaching and learning efficiency. SATCN not only outperforms the state-of-the-art models with the same type, but also solves ubiquitous problem of negative historical data. In future work, we will make efforts to improve model interpretability and performance by utilizing graph structure and more robust neural networks.

References

1. Khajah, M., Lindsey, R.V., Mozer, M.C.: How Deep is Knowledge Tracing?. In Proceedings of the 9th International Conference on Educational Data Mining, (EDM), Raleigh, North Carolina, USA, June 29 - July 2 (2016)
2. Corbett, A.T., Anderson, J.R.: Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction* **4**(4), (1994), 253–278 (1994)
3. Kuh, G.D., Kinzie, J., Buckley, J.A., Bridges, B.K., Hayek, J.C: Piecing together the student success puzzle: research, propositions, and recommendations: ASHE Higher Education Report. Vol. 116. John Wiley & Sons (2011)
4. Piech, C., Spencer, J., Huang, J., et al.: Deep knowledge tracing. arXiv preprint [arXiv:1506.05908](https://arxiv.org/abs/1506.05908) (2015)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
6. Zhang, J., Shi, X., King, I., Yeung, D.Y.: Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th International Conference on World Wide Web, pp. 765–774 (2017)
7. Abdelrahman, G., Wang, Q.: 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 (2019)
8. Santoro, A., Bartunov, S., Botvinick, M., et al.: Meta-learning with memory-augmented neural networks. In: International Conference on Machine Learning. PMLR, pp.1842–1850 (2016)
9. Polson, M.C., Richardson, J.J.: Foundations of intelligent tutoring systems. Psychology Press (2013)
10. Baker, R.S.J., Corbett, A.T., Aleven, V.: More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 406–415. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69132-7_44
11. Pardos, Z.A., Heffernan, N.T.: KT-IDEM: introducing item difficulty to the knowledge tracing model. In: Konstan, J.A., Conejo, R., Marzo, J.L., Oliver, N. (eds.) UMAP 2011. LNCS, vol. 6787, pp. 243–254. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22362-4_21
12. Pardos, Z.A., Heffernan, N.T.: Modeling individualization in a bayesian networks implementation of knowledge tracing. In: De Bra, P., Kobsa, A., Chin, D. (eds.) UMAP 2010. LNCS, vol. 6075, pp. 255–266. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13470-8_24
13. Yudelson, M.V., Koedinger, K.R., Gordon, G.J.: Individualized bayesian knowledge tracing models. In: Lane, H.C., Yacef, K., Mostow, J., Pavlik, P. (eds.) AIED 2013. LNCS (LNAI), vol. 7926, pp. 171–180. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39112-5_18
14. Ebbinghaus, H.: Memory: a contribution to experimental psychology. *Ann. Neurosci.* **20**(4), 155 (2013)
15. Pavlik Jr, P.I., Cen, H., Koedinger, K.R.: Performance factors analysis-a new alternative to knowledge tracing. Online Submission (2009)
16. Pandey, S., Karypis, G.: A Self-Attentive model for Knowledge Tracing. ArXiv preprint [arXiv:1907.06837](https://arxiv.org/abs/1907.06837) (2019)

17. Choi, Y.: Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing. ArXiv preprint [arXiv:2002.07033](https://arxiv.org/abs/2002.07033) (2020)
18. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
19. Nakagawa, H., Iwasawa, Y., Matsuo, Y.: Graph-based knowledge tracing: modeling student proficiency using graph neural network. In: 2019 IEEE/WIC/ACM International Conference On Web Intelligence (WI). IEEE, 2019: 156–163 (2019)
20. Yang, Y., Shen, J., Qu, Y., et al.: GIKT: A Graph-based Interaction Model for Knowledge Tracing (2020)
21. Gori, M., Monfardini, M., Scarselli, F.: A new model for learning in graph domains. In Neural Networks, 2005. In: IJCNN'05. Proceedings 2005 IEEE International Joint Conference on, Vol. 2. IEEE, 729–734 (2005)
22. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint [arXiv:1609.02907](https://arxiv.org/abs/1609.02907) (2016)
23. Kang, Y., Chen, Y., Chen, Y., et al.: Probabilistic forecasting with temporal convolutional neural network. *Neurocomputing* **399**, 491–501 (2020)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
25. Yeung, C.K., Yeung, D.Y.: Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In: Proceedings of the Fifth Annual ACM Conference on Learning at Scale, pp. 1–10 (2018)