Deep Knowledge Tracing with Convolutions

Shanghui Yang
zqws1018@outlook.com
East China Normal University
Shanghai, China

Mengxia Zhu
51195100046@stu.ecnu.edu.cn
East China Normal University
Shanghai, China

Jingyang Hou
51195100033@stu.ecnu.edu.cn
East China Normal University
Shanghai, China

Xuesong Lu
xslu@dase.ecnu.edu.cn
East China Normal University
Shanghai, China

ABSTRACT

Knowledge tracing (KT) has recently been an active research area of computational pedagogy. The task is to model students' mastery level of knowledge based on their responses to the questions in the past, as well as predict the probabilities that they correctly answer subsequent questions in the future. A good KT model can not only make students timely aware of their knowledge states, but also help teachers develop better personalized teaching plans for students. KT tasks were historically solved using statistical modeling methods such as Bayesian inference and factor analysis, but recent advances in deep learning have led to the successive proposals that leverage deep neural networks, including long short-term memory networks, memory-augmented networks and self-attention networks. While those deep models demonstrate superior performance over the traditional approaches, they all neglect more or less the impact on knowledge states of the most recent questions answered by students. The forgetting curve theory states that human memory retention declines over time, therefore knowledge states should be mostly affected by the recent questions. Based on this observation, we propose a Convolutional Knowledge Tracing (CKT) model in this paper. In addition to modeling the long-term effect of the entire question-answer sequence, CKT also strengthens the short-term effect of recent questions using 3D convolutions, thereby effectively modeling the forgetting curve in the learning process. Extensive experiments show that CKT achieves the new state-of-the-art in predicting students' performance compared with existing models. Using CKT, we gain 1.55 and 2.03 improvements in terms of AUC over DKT and DKVMN respectively, on the ASSISTments2009 dataset. And on the ASSISTments2015 dataset, the corresponding improvements are 1.01 and 1.96 respectively.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Applied computing** → **Learning management systems**.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

Deep knowledge tracing, Convolution neural network, Sequence modelling, Education data mining

ACM Reference Format:
Shanghui Yang, Mengxia Zhu, Jingyang Hou, and Xuesong Lu. 2020. Deep Knowledge Tracing with Convolutions. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

One important aspect in education is to continuously estimate each student's mastery level of knowledge that is taught, or simply termed knowledge state. According to the evolution of one's knowledge state, a tutor may properly design a personalized learning path for the student, until she masters all the knowledge. Among those alternative estimation methods, *knowledge tracing* (KT) models students' changing knowledge state by tracing their interactions with coursework, i.e., a sequence of questions being solved. By observing whether a student correctly answer each successive question containing a particular knowledge concept, the model can adjust her overall knowledge state and also predict her performance on the next question. Thanks to the ease of interpretation and adoption, knowledge tracing has been widely used in intelligent tutoring systems and recently in MOOC platforms [**? ?**].

Knowledge tracing models are used to be constructed using statistical modeling methods such as Bayesian inference with a Hidden Markov model [8, 33] and factor analysis with logistic regression [3, 5, 22]. Due to the availability of massive educational data released by large MOOC platforms and educational institutions, a recent trend is to train neural network based models for knowledge tracing, which have shown superior performance over traditional methods. In the pioneering work [23], Piech et al. propose the DKT model using an LSTM network and significantly improve the overall AUC of predicting students' performance on solving the next questions. The model reads each student-question interaction (consisting of a question and the correctness of the student's answer) sequentially and predicts whether the student answer the next question correctly. Inspired by this work, a series of deep learning models are proposed to target various aspects in the knowledge tracing task, including DKVMN [34], EERNN [28], EKT [14], SKVMN [1] and SAKT [21], etc. Readers may refer to Section 2 for a detailed literature review of existing KT models.

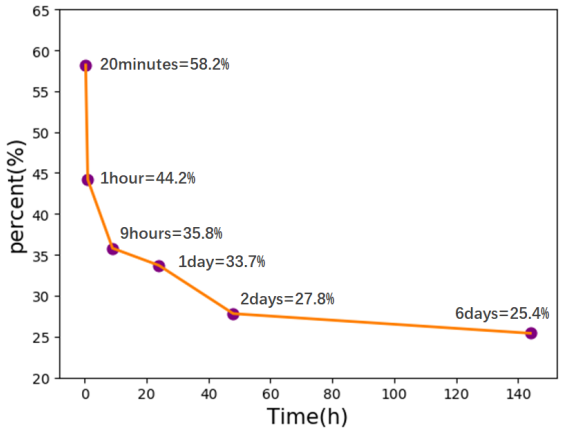

Figure 1: The forgetting curve describes the law of the human brain forgetting new things. This curve has a major impact on the study of human memory cognition. It reveals that forgetting begins immediately after learning, when only 20 minutes have passed, only 58.2% of memory remains. Therefore, recent exercises play a vital role in human memory.

Despite their superiority in predicting student performance, we notice two major defects in the existing models. First, existing methods always rely on the entire student-question interaction sequence in the past to model knowledge state, that is, they have not given extra consideration to the recent interactions in the model. This is somehow counter-intuitive because a question answered in a very early stage should have trivial effect on the question being solved at present. The theory of Ebbinghaus Forgetting Curve [9] states that human memory retention of knowledge declines very quickly if there is no attempt to retain it, as shown in Figure 1. We can see that in 20 minutes after learning, only 58% of memory is retained, and this number drops to 25.4% in 6 days. Therefore, we speculate that the effect of recently answered questions should be strengthened in order to better model students' knowledge state.

Secondly, all existing methods embed a student-question interaction into a vector and directly feed the embedded interaction sequence into models. A commonly used setting is to represent a question and the answer of a student using a one-hot encodings. Such a naive representation cannot leverage network structures that extract important local patterns, and thus may limit the predictive power of the models. We attempt to solve this problem by reshaping the vectorized embedding into a matrix and then using convolutions to extract its local patterns. These convolutions can extract important spatial patterns from the student-question interactions, which would be not possible with only LSTM units.

Motivated by these two observations, we propose in this work a new KT model, which is called *Convolution-augmented Knowledge Tracing*, or simply CKT. At the core of the new architecture is a 3D convolution network for capturing the information from the recent student-question interactions. In particular, given a student-question interaction sequence, we first embed each interaction into a vector and feed them sequentially into a classic LSTM network, following the method in DKT [23]. The LSTM network outputs a hidden representation at every step t, which represents the hidden state of the processed interactions in the sequence so far. Meanwhile, at every step t we reshape each of the previous k embeddings, i.e., the embeddings from step $t-k+1$ to t, into a matrix (feature map) preserving the same elements in the corresponding vector embedding, and then stack the k matrices in their original chronological order to form a three-dimensional tensor. The tensor thus represents the information of very recent interactions for which the student still have relatively high memory retention. Then motivated by video recognition tasks, we employ a 3D convolutional network [30, 31] to learn from the tensor, which eventually outputs a hidden representation with length equal to that of the hidden representation generated by the LSTM network. The benefit of using 3D convolutions is two-fold. First, the convolutions can extract important spatial patterns from the reshaped matrix. Second, the 3D convolutions can establish the relationship on three dimensions, thus correlating the features over the stacked matrices. Next, we borrow the idea of the threshold mechanism in GRU [7] and propose an update gate to fuse the two hidden representations. Finally, the fused representation is transformed to predict the student's mastery level of each knowledge concept. We use an additional LSTM network in the main model for the transformation of the fused representation, and show this component can improve the prediction performance in the ablation study. Experimental results demonstrate that CKT outperforms main existing KT models in predicting student performance, and hence can be used to better model student knowledge state.

To summarize, our contribution in this work is four-fold.

(1) We reveal the importance of the recent student-question interactions in the knowledge tracing task. We devise a novel model CKT to collectively learn the long-term feature of the entire interaction sequence and the short-term feature of the most recent interactions, and show the short-term feature can strengthen the predictive power of the model.
(2) We demonstrate that using a 3D convolutional architecture to learn from the reshaped embeddings of the recent student-question interactions can improve the prediction performance. To the best of our knowledge, we are the first to introduce a convolutional network in the knowledge tracing task. The overall architecture not only achieves the new state-of-the-art performance, but also serves as a general framework for similar prediction problems with time series data, i.e., using 3D convolutions as feature extractors to improve model performance.
(3) To fuse the long-term and the short-term features, we borrow the idea from the threshold mechanism in GRU and propose an update gate for adaptively fusing these two features.
(4) We conduct extensive experiments to demonstrate the effectiveness of the CKT model. Compared with existing Deep Learning (DL)-based models of diverse architectures, CKT achieves the highest AUC when predicting whether students can correctly answer the next questions. We also conduct

sensitivity analysis and ablation study to show the rationality of hyperparameter settings and the usefulness of model components.

The rest of this paper is organized as follows. In Section 2 we present a comprehensive literature review of existing DL-based knowledge tracing models and introduce some basics pertaining to 3D convolutions. In Section 3 we formally define the knowledge tracing problem. We describe the architecture details of the proposed CKT model in Section 4, followed by extensive performance evaluation in Section 5. We discuss two popular KT applications using CKT in Section 6 and finally conclude in Section 7.

2 RELATED WORK

2.1 Knowledge Tracing

The idea of knowledge tracing is proposed in [8], where the authors construct a tutoring system with a production rule cognitive model of programming knowledge concepts. As a student solves programming questions, the system estimates the probability that the student has learned each concept, i.e., estimating the student's programming knowledge state. They propose a tracing model called Bayesian Knowledge Tracing (BKT). The model uses four parameters for each knowledge concept and employs a Hidden Markov model with Bayesian inference to fit the sequence data of student-question interaction on the learning system. BKT assumes that the knowledge state at the present step only depends on the state at the previous step and thus has very limited ability for knowledge tracing. Also, one has to build an independent model for each knowledge concept. Later, a series of factor analysis models using logistic regression are proposed [3, 5, 22]. These models manually construct input features such as the number of attempts for each question, the number of correct and incorrect attempts and the number of mentions for each knowledge concept. The results show these simple models have predictive power comparable to BKT.

2.2 Knowledge Tracing with Deep Learning

The trend of using deep learning for the KT task is pioneered by the work of DKT [23]. DKT uses an LSTM network [10, 29] to learn from the student-question interaction sequences. At each step, the LSTM unit takes as input an interaction tuple representing which question is answered and whether the answer is correct. The tuple is simply encoded using a one-hot vector. The output is a vector of length equal to the number of questions, where each element represents the predicted probability that the student would correctly answer the particular question (knowledge state). The output vector is transformed into a prediction vector pertaining to the next question, which predicts whether the next question is answered correctly. Finally, the loss function is computed using binary cross entropy between the predicted responses and the ground-truth responses. The result of DKT is significantly higher than BKT and its variants in terms of AUC.

Then inspired by Memory-Augmented Neural Networks [11, 25], Zhang et al. [34] propose Dynamic Key-Value Memory Networks (DKVMN) to improve DKT's structure. The assumption is that the hidden state in the LSTM network has limited power to represent hidden knowledge state, therefore memory matrices should be introduced to store more abundant hidden information. DKVMN uses two memory matrices, where one is static and used to store the latent knowledge concepts of the questions, and the other is dynamic and used to represent the student's knowledge state, each matrix slot storing the state of one concept. DKVMN updates the knowledge state of a student by reading and writing to the dynamic matrix using correlation weights computed from the input question and the static matrix. Following this work, Abdelrahman et al. propose Sequential Key-Value Memory Networks (SKVMN) [1] to capture the dependencies between questions. They assume that the predicted result of the next question only depends on the previous interactions pertaining to the questions with the same concept. Therefore at each step, they introduce an additional hop-LSTM network before the output layer of DKVMN, whose LSTM units connect only the hidden states of the steps pertaining to the dependent questions (picked using triangular membership function [15]). The output of the hop-LSTM network is used to predict the response of the next question.

Another line of work attempts to incorporate additional features as model input. For example, EERNN [28] uses a Bi-LSTM network to obtain the text embedding of each question, and concatenates the embedding with that of the corresponding student-question interaction tuple as used in DKT. The concatenated embeddings now contain the text feature of the questions and are fed into a LSTM network. EERNN employs two different architectures to represent the hidden state of the past interaction sequence. One uses the hidden state of the last LSTM unit as in DKT. The other uses an attention mechanism to aggregate all the hidden states of the past LSTM units. Experimental results show the attention mechanism brings additional boost on prediction performance. Later, Huang et al. [14] extend EERNN and propose the EKT model, which borrows from the idea of memory networks and replaces the hidden state in the LSTM network with a hidden matrix. Other work of this line includes DKT+forgetting [18] that manually constructs features pertaining to forgetting behaviors, DKT-DSC [17] that clusters students in every few steps and uses the clustering results as additional input, PDKT-C [4] that incorporates prerequisite relations between knowledge concepts as additional constraints, etc.

There are other efforts that use more recently proposed architectures. For example, Pandey et al. [21] propose the model of Self-Attentive Knowledge Tracing (SAKT), with the hope to handle the data sparsity problem by using the $Transformer$ architecture [32]. When predicting the response to the next question, SAKT attends to all the previous student-question interactions by assigning a learnable weight to each of them. Nakagawa et al. [19] incorporate Graph Neural Networks into their KT model and propose Graph-based Knowledge Tracing (GKT). They construct a graph such that the nodes are the knowledge concepts and there is an edge between two nodes if the corresponding concepts have dependency relationships. When a student answers a question associated with a particular concept, GKT first aggregates the node features related to the answered concept, and then updates simultaneously the studentâĂŹs knowledge state for the answered concept itself as well as for the related concepts. The updating process employs a multilayer perceptron (MLP) layer, an eraseâĂŞadd gate used in DKVMN [34], and a gated recurrent unit (GRU) [6] gate.

Knowledge tracing with deep learning has shown promising performance in capturing knowledge state and predicting future

behaviors. However, all aforementioned models have not considered the biased impact of the most recent student-question interactions. Below we show that strengthening such impact in the model indeed boosts significantly the prediction performance.

2.3 3D Convolution

Because of its rich nonlinearity and excellent performance, convolution neural networks have become very popular in the field of computer vision, such as object detection [24], image segmentation [12] and optical character recognition (OCR) [26]. Inspired by image, 3D convolution [30, 31] (3D Conv) was proposed to handle video analysis tasks. A video can be viewed as a sequence of images, it has one more time dimension than the single image, and video data can be formulated as a (D × H × W) tensor, D represents the depth or time of video, H and W represent the height and width of one frame image, respectively. Tran et al. [30] has demonstrated that compared to 2D Conv, 3D Conv has ability to model temporal information better, because in 3D Conv, convolution operations are performed spatio temporally while in 2D Conv they are done only spatially. One 2D Conv applied to multiple images will result in only one image, making 2D Conv lose temporal information of the input signal right after every convolution operation. But 3D Conv will preserve temporal information during the convolution operation.

We believe that since 3D Conv can capture the temporal information of video, it should have the ability to capture the temporal information of general time-series data such as a sequence of student-question interactions, so we propose a group of 3D Conv to model the time-series relationship between interaction sequences.

In the field of knowledge tracing, existing models encoded a student-question interaction as a single vector, and directly feed it into models. Such a navie representation cannot utilize the adavantages of neural networks and cannot model students' knowledge states effectively, because a single vector input lose important spatial patterns from the student-question interactions. We attempt to solve this problem by reshaping the embedding vector into a matrix and then using convolutions to capture the spatial patterns from matrix input, which would be not possible with only LSTM units.

3 PROBLEM DEFINITION

Knowledge tracing takes as input a sequence of student-question interaction and outputs the prediction of the student's response to the next question. As such it can be viewed as a time series prediction problem and an auto-regressive problem. Formally, the problem of knowledge tracing can be defined as follows.

Definition of (deep) knowledge tracing. *For each student, denote by q_i the i^{th} question he answers and by a_i the corresponding response of the student. Given a sequence of student-question interactions $\mathcal{X} = \{x_1, x_2, \ldots, x_t\}$, where $x_i = (q_i, a_i)$, KT predicts the student's response a_{t+1} to the next question q_{t+1}, i.e., the probability* $\mathrm{P}(a_{t+1} = 1 | q_{t+1}, \mathcal{X})$ *that the student answers the next question correctly.*

In the above definition, a_i is a binary variable where 1 represents the student's answer is correct and 0 otherwise. In practice, x_i is encoded as a one-hot vector of length $2M$, where M is the number of questions. If the answer is incorrect, the element of x_i in the first half of the vector corresponding to the question q_i has value 1; otherwise, the corresponding element in the second half of the vector has value 1.

4 THE CKT MODEL

In this section, we describe the detail architecture of the CKT model. We first give an overview of CKT, whose main components consist of an LSTM network to capture the long-term feature from the entire sequence of interactions, and a 3D convolutaional network to capture the short-term feature from the most recent interactions. Then we describe the structure of the proposed 3D convolutional network, followed by the proposal of the update gate that fuses adaptively the two types of features. Finally, we introduce the objective function used in CKT.

4.1 Model Overview

Figure 2 illustrates the architecture of CKT. In the original input sequence $\mathcal{X} = \{x_1, x_2, \ldots, x_t\}$, each x_i is a one-hot vector of length $2M$, as described in the last paragraph of Section 3. To overcome the sparsity issue and facilitate the convolutional layers, we first use an embedding layer (which is omitted in the figure) to convert each input token x_i into a dense embedding $\hat{x}_i$ with length d_e. Then we use a classic LSTM network as used in DKT to learn from the entire sequence $\hat{\mathcal{X}} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_t\}$, but do not directly use the hidden states for prediction, that is, at step t the LSTM network only outputs a hidden state vector h_t of length d_h.

In parallel with the LSTM network, we employ a 3D convolutional network to extract the short-term feature in the sequence. In particular, we reshape each of the k most recent dense embeddings $\hat{x}_i$ for $i \in [t-k+1, t]$ into a matrix (feature map) with shape $H \times W$, where $H \times W = d_e$. In CKT, we set $H = W$, but one may set it to any shape as long as the equation holds. Then we stack the k matrices in their original chronological order and form a three-dimensional tensor of shape $k \times H \times W$, where k is the depth of the tensor. The 3D convolutional network accepts the tensor as input and outputs a tensor with the same shape. After a global average pooling layer, the output tensor is squashed on the time dimension into a matrix of shape $H \times W$, which is further stretched into a hidden vector $\hat{m}_t$ of size d_e. In practice, we set $d_e = d_h$, the depth k and the embedding size d_e (or matrix heigh H) are the two hyperparameters to be adjusted.

Now we have two hidden state vectors h_t and $\hat{m}_t$, representing the long-term feature and the short-term feature, respectively. In order to balance the two features, we borrow the idea of the threshold mechanism of GRU [7] and propose an update gate to adaptively fuse the two features. The update gate outputs a final hidden state $\tilde{h}_t$ of step t.

At Last, an additional LSTM layer is employed to transform $\tilde{h}_t$ and output a vector y_t with length M, where each element represents the probability that the student has mastered the corresponding knowledge concept at time t. Here one may also use a fully-connected layer instead, but we will show the LSTM layer

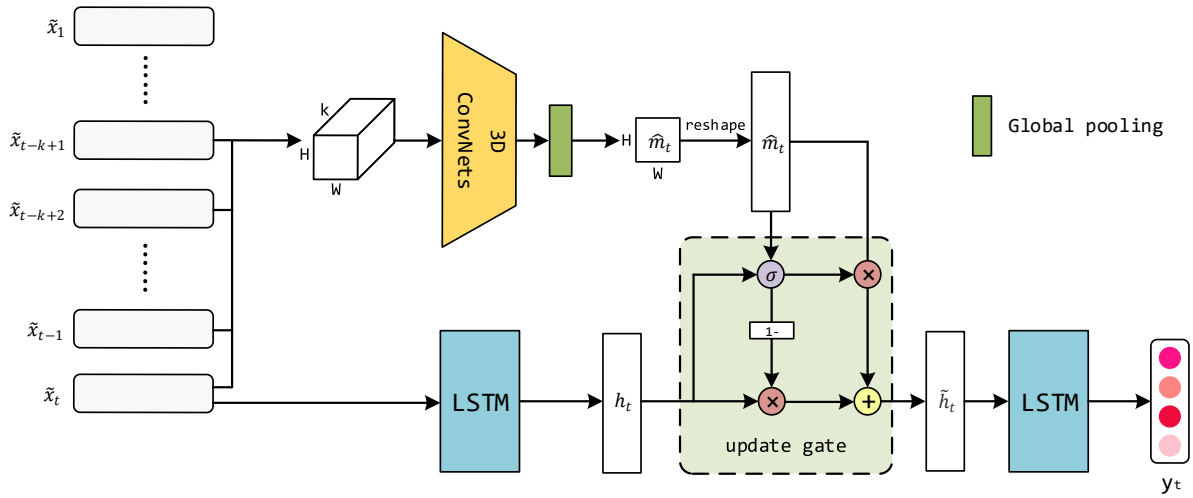

Figure 2: The architecture of our CKT model, two submodules (3D ConvNets and LSTM) feed forward in parallel during training and testing, followed by an update gate to adaptively fuse two features from two submodules, and an LSTM processes the fused features, outputs final prediction at the end of model.

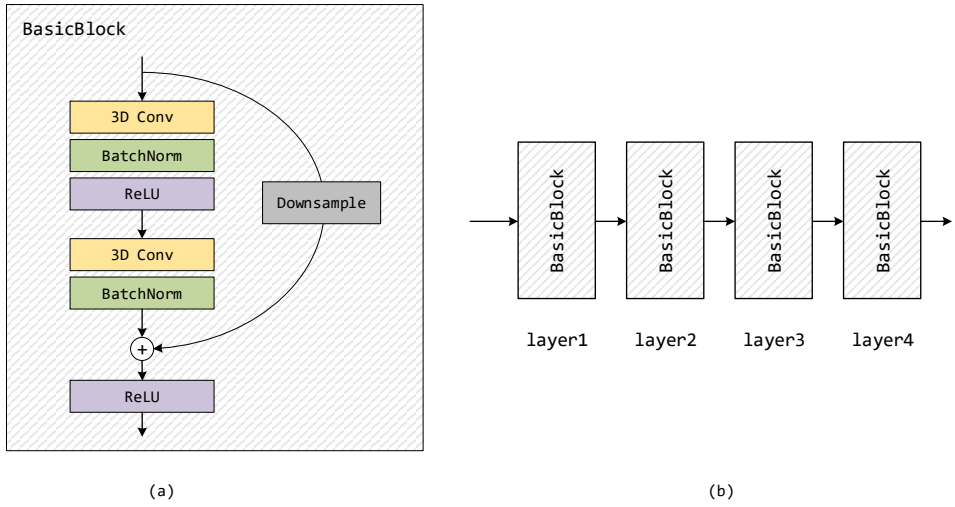

Figure 3: The model architecture of 3D convolution Networks (3D ConvNets) module, it consists of 4 stacked BasicBlocks as shown in (b) and each BasicBlock is a residual architecture with 3D convolution which is shown in (a). Detail parameters are presented at Table 1.

generates better results. The predicted response to the next question q_{t+1} can be calculated by directly indexing from output vector in q_{t+1} th element.

4.2 The 3D Convolutional Network

The 3D convolutional network (3D ConvNets) takes as input a three-dimensional tensor with shape $k \times H \times W$, as described in Section 4.1, which wraps the information in the k most recent interactions. In case the current step t is less than k, we pad the tensor with zero matrices until the depth equals k.

The architecture of the 3D ConvNets is shown in Figure 3. We design a block named BasicBlock as shown in Figure 3(a), and stack four BasicBlocks in CKT as shown in Figure 3(b). The BasicBlock consists of two 3D convolutional layers, each of which followed by

Table 1: Detailed parameters of 3D ConvNets Module

layer name	output size	kernel size, in channel, out channel
layer1	$k \times \mathrm{H} \times \mathrm{W}$	$\begin{bmatrix} 3\times3\times3, (1,4) \\ 3\times3\times3, (4,4) \end{bmatrix}$
layer2	$k \times \mathrm{H} \times \mathrm{W}$	$\begin{bmatrix} 3\times3\times3, (4,8) \\ 3\times3\times3, (8,8) \end{bmatrix}$
layer3	$k \times \mathrm{H} \times \mathrm{W}$	$\begin{bmatrix} 3\times3\times3, (8,4) \\ 3\times3\times3, (4,4) \end{bmatrix}$
layer4	$k \times \mathrm{H} \times \mathrm{W}$	$\begin{bmatrix} 3\times3\times3, (4,1) \\ 3\times3\times3, (1,1) \end{bmatrix}$
global pooling	$\mathrm{H} \times \mathrm{W}$	-

a batch normalization layer and a ReLU layer. In addition, we use a shortcut connection to sum the input of the BasicBlock with the output of the second batch normalization layer, before the second ReLU layer. This forces the convolutional network to learn the residual features [13] from the tensors and facilitates the network optimization. In order to keep the size of input and output feature maps consistent, similar to the ResNet [13], we use a small 3D convolution with kernel size $1 \times 1 \times 1$ and reshape the input to the same size of output, which is called Downsample in Figure 3(a). In total, the BasicBlock can be formulated as follows:

$$T_{i+1} = ReLU(BN(Conv(ReLU(BN(Conv(T_i))))) + Down(T_i)) \quad (1)$$

where T_i represents input tensor, $Conv$, BN, $ReLU$, $Down$ represent the calculations at the 3D convolutional layer, the batch normalization layer, the ReLU layer and the function of Downsample, respectively. T_{i+1} represents the output tensor with the same shape as input T_i.

There are 4 BasicBlocks in 3D ConvNets module which is shown in Figure 3 (b). Following the principles of VGG [27] and FCN [16], a small filter should be used instead of a large filter and pooling layers can be completely replaced by convolutions. We set all 3D convolutions have $3 \times 3 \times 3$ kernel size with 1 stride and 1 zero padding, which guarantees the shape invariance mentioned above. Detail parameters are presented in Table 1.

A global average pooling layer accepts the 3D ConvNets output tensor T_4, i.e. the output of fourth BasicBlock, and squashes the tensor into a matrix m_t of shape $H \times W$ on the time dimension, which can be formulated as follows:

$$m_t = \frac{1}{K} \sum_{k=1}^{K} m^k, \quad (2)$$

where K is the number of recent student-question interactions, m^k is the k th feature map in T_4. The output matrix m_t has the shape of $H \times W$. Then the output matrix is further stretched into a hidden vector $\hat{m}_t$ of size d_e. In our setting, $d_e = d_h$, so that long-term features h_t and short-term features $\hat{m}_t$ have the same length.

4.3 Adaptive Feature Fusion

In order to fuse long-term features h_t and short-term features $\hat{m}_t$, we borrow the idea from the threshold mechanism in LSTM [10] and GRU [7] and propose an update gate mechanism for adaptively learning the weights of these two features, the weights control how much information from two features will be preserved. The process can be formulated as follows:

$$\text{update gate}: \ \mathbf{z}_t = \sigma([\hat{\mathbf{m}}_t, \mathbf{h}_t]\,\mathbf{W}_z + \mathbf{b}_z) \quad (3)$$

$$\tilde{\mathbf{h}}_t = \mathbf{z}_t \odot \hat{\mathbf{m}}_t + (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h_t} \quad (4)$$

where $[\cdot, \cdot]$ represents concatenating two vectors, $\mathbf{W}_z$ is the weight matrix of fully connected layer, which has a shape of $2d_h \times d_h$, $\mathbf{b}_z$ is the bias vector of fully connected layer, which has a shape of d_h, σ represents the sigmoid function, $\odot$ means the Hadamard product between two vectors.

We concatenate $\hat{m}_t$ with h_t into a vector with length $2d_h$, and then send it to a fully connected layer with output size d_h followed by a sigmoid activation function, which is shown in equation 3. We get $\mathbf{z}_t$ as update gate to control the ratios of $\hat{m}_t$ and h_t, then fuse two features based on the $\mathbf{z}_t$, which is shown in equation 4.

Finally, an additional LSTM layer is employed to transform $\tilde{\mathbf{h}}_t$ and predicts a probability vector y_t with length M, where each element represents the probability that the student has mastered the corresponding knowledge concept at time t. The predicted response to the next question q_{t+1} can be calculated by directly indexing from output vector the corresponding element.

4.4 Objective Function

The objective function we used is a binary cross-entropy loss function, calculated between the predicted probability p_t and ground truth answer a_t for all time step t, where p_t is the probability of correctly answering the question q_t. The function can be formulated as:

$$\mathcal{L} = -\sum_t (\mathrm{a}_t \log \mathrm{p}_t + (1 - \mathrm{a}_t) \log(1 - \mathrm{p}_t)) \quad (5)$$

Table 2: Statistics of dataset

Dataset	#Questions	#Students	#Exercises	#Exercises per student
ASSISTments2009	110	4,151	325,637	78
ASSISTments2015	100	19,840	683,801	34
Statics2011	1,223	333	189,297	568
Synthetic-5	50	4,000	200,000	50

5 EXPERIMENTS

5.1 Datasets

We use four datasets to evaluate our CKT model, the statistics of datasets are shown in Table 2.

Table 3: The results of our CKT model and others, we choose the best test AUC results (%) during training process of these models on 4 datasets. The bolded AUC is the best result in each dataset, and the AUC with * is the best result of CKT. It can be found that our CKT model performs better than other models on all datasets except for Synthetic-5, because this dataset is synthetic and doesn't accord with the normal learning process of human beings.

Dataset	DKT	DKVMN	SKVMN	SAKT	CKT			
					k=4 b=80	k=8 b=64	k=16 b=48	k=32 b=32
ASSISTments2009	80.99	80.51	68.48	75.41	**82.54***	82.41	82.10	81.51
ASSISTments2015	71.90	70.95	66.82	71.85	**72.91***	72.81	72.85	72.74
Statics2011	82.79	79.53	78.40	81.34	**82.41***	82.10	82.09	81.80
Synthetic-5	80.71	**82.53**	78.05	82.33	82.43*	82.15	82.05	82.25

- **ASSISTments2009**[1] : This dataset is gathered in the school year 2009-2010 from the ASSISTments education platform[2]. We use the skill builder data of ASSISTments2009, it consists of 110 distinct questions, 4, 151 students and a total number of 325, 637 exercise records (student-question interactions).
- **ASSISTments2015**[3] : This dataset was collected in 2015. This is an update to the ASSISTments2009. It includes 100 distinct questions, 19, 840 students and a total of 683, 801 exercise records. This dataset has the largest number of students, but the average number of exercises per student is low.
- **Statics2011**[4] : This dataset was collected from a statistics course at Carnegie Mellon University in the fall of 2011. It contains 1,223 distinct questions, 333 students, a total of 189,297 exercise records.
- **Synthetic − 5**[5] : This dataset is generated by Piech et al. [23]. It consists of training data and testing data. Each set contains 50 distinct questions, 4,000 virtual students and 200,000 exercise records.

Among these four datasets, the two datasets from ASSISTments are mostly suitable for evaluating KT models because they have appropriate sequence length and number of training sequences. Statics2011 has too few number of sequences, which cannot even satisfy the size of one batch. Also, the sequence length is often too long to be realistic. So following the method in other work [23, 34], we take a fold operation on this dataset. In particular, when a sequence length exceeds 200, we truncate the sequence and put the extra interaction data into the next sequence. Thus all the sequences have length less than or equal to 200. Synthetic-5 is a synthetic dataset, which is used for benchmark purpose. The details of synthesis can be found in the paper [23].

[1]ASSISTments2009:https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010
[2]https://www.assistments.org/
[3]ASSISTments2015:https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data
[4]Statics2011:https://pslcdatashop.web.cmu.edu/ DatasetInfo?datasetId=507
[5]Synthetic-5:https://github.com/chrispiech/DeepKnowledgeTracing/tree/master/data/synthetic

5.2 Network Instance

The overview of CKT model is shown in Figure 2, we detail the hyper-parameters we used here. We set both d_h and d_e to 225, which represents hidden size of LSTM and embedding size respectively. The first LSTM and second LSTM use the same hidden size and both have just one layer. H and W are therefore equal to 15, so that $H \times W = d_e$. Parameter k is also adjustable. We set $k = 4$ in the main results. We also perform a sensitivity analysis on the hyperparameters in Section 5.5.

The 3D ConvNets is constructed by stacking four BasicBlocks, each of which contains two Conv-BN-ReLU layers. The complete settings of 3D ConvNets are shown in Table 1. We set all kernel size of convolutions to $3 \times 3 \times 3$, and vary their channel sizes in the forward pass. The size of the BN and ReLU layers are omitted in the table.

5.3 Training

We implement CKT using Pytorch1.4 with CUDA 10.2 on NVIDIA Titan RTX GPU. The Adam optimizer is used to optimize the model. Following the rule of Occam's Razor, we set the L2 regularization term to 1e-5. The initial learning rate is 0.001, and learning rate decay is set to 0.3 every 5 epochs, we trained 20 epochs in total.

We use default division of train sets and test sets provided by the datasets' creators. We train all the models on the train data and evaluate them on test data. The metrics we use are the area under the Receiver Operation Characteristic (ROC) curve, referred to as AUC [2]. When AUC = 0.5, it means that the prediction of model is as same as random guess. The higher value of AUC, the better the performance of the model.

5.4 Comparative Experiments

We explore several different combinations of k and batch size b. Limited by GPU memory, increasing k will increase the memory utilization during training, so we have to reduce the batch size b to prevent from out-of-memory error. Besides, we fix hyper-parameter $H = 15$ (and thus $d_e = 225$), i.e. the height of feature map in 3D ConvNets. In section 5.5, we will conduct some sensitive experiments on these three hyper-parameters for exploring the effects of different parameters on the prediction performance. In Table 3, best

Table 4: The AUC results (%) of sensitive experiments for parameter k, we fix the batch size $b = 64$ and $H = 15$, and adjust k for exploring the effect of k on the model. Best AUC is bolded.

b&H	k	ASSIST2009	ASSIST2015	Statics	Synthetic
b=64 H=15	k=2	82.21	72.97	82.13	**82.42**
	k=4	82.39	72.81	82.11	82.31
	k=6	82.30	72.81	**82.23**	82.22
	k=8	**82.41**	72.81	82.10	82.15
	k=10	82.15	72.84	81.07	82.12
	k=12	82.13	72.86	80.40	82.12
	k=14	81.48	73.02	80.52	82.11
	k=16	81.58	**73.14**	80.02	82.08

Table 5: The AUC results (%) of sensitive experiments for parameter b, we fix the $k = 4$ and $H = 15$, and adjust b for exploring the effect of batch size on the model. Best AUC is bolded.

k&H	b	ASSIST2009	ASSIST2015	Statics	Synthetic
k=4 H=15	b=8	80.91	71.83	81.82	81.92
	b=16	81.71	72.59	82.24	82.18
	b=32	82.00	72.71	82.28	82.22
	b=48	82.51	72.60	**82.52**	82.27
	b=64	82.28	72.82	82.39	82.32
	b=80	**82.54**	72.91	82.41	82.43
	b=96	82.25	**72.92**	81.39	**82.43**

results for each dataset have been bolded, we can find that $k = 4$ and $b = 80$ yield the best AUC.

We also compare with several state-of-the-art models in knowledge tracing, namely, DKT [23], DKVMN [34], SKVMN [1], SAKT [21]. We don't have datasets with question's text. Therefore the result of EERNN and EKT cannot be implemented. We trained these methods with our CKT both on same train datasets, and tested these models on same test datasets, we get the best test AUC as report scores in Table 3. Our CKT model has considerable advantages and is promising compared with other models.

We reproduce DKT, DKVMN and SKVMN based on their paper, and get the code of SAKT from author of [21], we find that there is a fatal bug in their code. When they padded the sequence with zeros for changing the variable length sequence to fixed length sequence, they didn't remove those zeros on measurement stage (i.e. calculate loss and AUC), result in a high AUC in their paper because zeros are also used as an exercise and answer to participate in the calculation. We remove those zeros on measurement stage.

5.5 Sensitive Experiments

In this part, we explore the influence of three hyper-parameters on our CKT model, we conduct sensitive experiments on the number of recent interactions k, batch size b, feature map height H, respectively. For each experiment, we fix two other parameters and adjust the remaining parameter to explore the sensitivity of the model.

Table 6: The AUC results (%) of sensitive experiments for parameter H, we fix the batch size $k = 4$ and $b = 64$, and adjust H for exploring the effect of H on the model. Best AUC is bolded.

k&b	H	ASSIST2009	ASSIST2015	Statics	Synthetic
k=4 b=64	H=11	82.16	**72.93**	81.03	82.09
	H=13	82.30	72.75	82.26	82.23
	H=15	**82.43**	72.79	82.41	82.33
	H=17	82.37	72.58	82.73	82.38
	H=19	82.19	72.68	**83.19**	**82.41**

Table 7: The AUC results (%) of ablation study, we propose 3 ablation models to verify the function of our model. The parameters of these models are same, we set $k = 4, b = 80, H = 15$. TOTAL_CKT performs better than others in all four datasets.

Ablation	ASSIST2009	ASSIST2015	Statics	Synthetic
LSTM_RECENT	81.63	72.71	79.66	81.07
FC_POOLING	81.86	72.69	82.23	82.02
FC_REAR	81.21	72.44	82.21	80.23
TOTAL_CKT	**82.54**	**72.91**	**82.41**	**82.45**

The first experiment focuses on k, we fix $b = 64$ and $H = 15$, remains other parameters unchanged. Results of best test AUC over four datasets can be found in Table 4. We can find that most datasets tend to have a better performance with a small k, it is in line with our expectations: the recent student-question interactions are important for modeling the knowledge states of students. Nevertheless, the value of k should be moderate for certain dataset, too high or too low of k will result in bad performance.

The second experiment focuses on b, we fix $k = 4$ and $H = 15$, try to use different batch size and train from scratch on each dataset. As shown in Table 5, in ASSIST2009, ASSIST2015 and Synthetic datasets, the larger batch size, the better performance. However, this trend is not shown in dataset Statics2011, we think that Statics2011 has total 333 records, and the number of train subset is even less, a big batch size will reduce efficiency of gradient descent.

The third experiment focuses on H, we fix $k = 4$ and $b = 64$, H represents the height of feature map in 3D ConvNets, and in CKT $H = W$, so H directly represents the size of feature map, i.e. the hidden size of our CKT model. For example, when $H = 11$, $d_e = d_h = H \times W = 121$, d_e and d_h are the length of two hidden vectors, respectively. The results can be found in Table 6, we can find that changes in H can fine-tune the model, but will not bring much improvement.

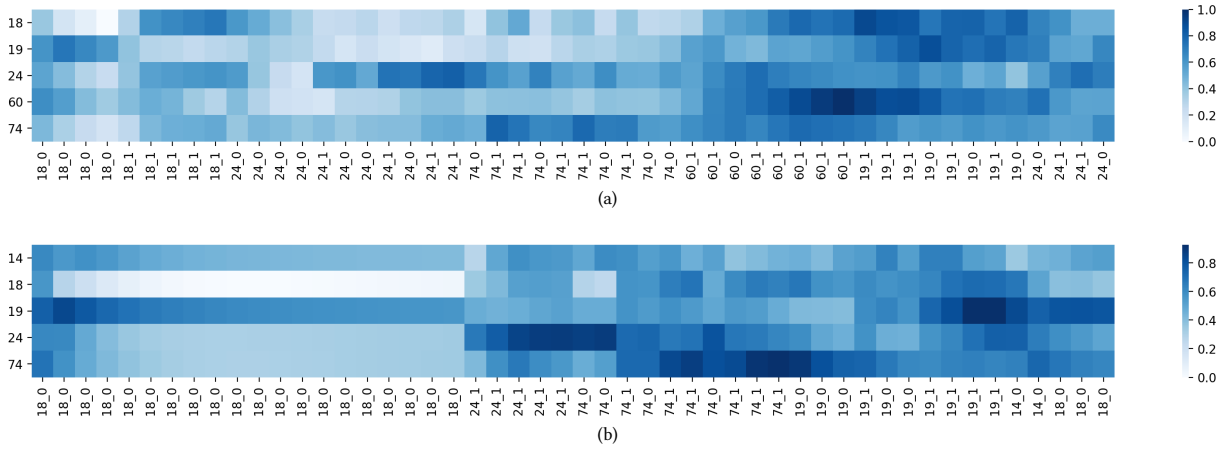

Figure 4: The evolution of knowledge states, y-axis represents question id, and x-axis represents student's interactions, each interaction is formulated by $q_t\_a_t$, q_t represents the question id answered at time t, and a_t represents the correctness of this answer, 1 represents answer is correct, 0 represents answer is incorrect.

5.6 Ablation Experiments

We conduct 3 ablation experiments, the results can be found in Table 7. Firstly, we consider that LSTM can be used to capture short-term features from recent interactions, we replace 3D ConvNets with an LSTM, named **LSTM_RECENT**. Secondly, global pooling is a simple and effective method to reduce dimension, for exploring the role of global pooling, we use a fully connected layer whose input dimension is $k \times H \times W$ and output dimension is $H \times W$ to replace global pooling, we call such an experiment as **FC_POOLING**. Thirdly, second LSTM in CKT is used to output final prediction, we use a fully connected layer named **FC_REAR** to accomplish the same function, the input is a vector with length d_h, and the output is a final prediction vector with length M. In all ablation experiments, we only change the submodule that we consider to ablate and keep other parameters unchanged. The results of **TOTAL_CKT** are higher than other ablation models over all four datases, it prove that all three submodules are important to CKT.

6 APPLICATION

6.1 Evolution of Knowledge States

The most direct application of knowledge tracing is to trace the evolution of knowledge states in the learning process of a student. It can help students and teachers understand the learning status of every student. We show two samples in Figure 4. The elements in matrices represent the knowledge state for a certain question, which are collected from prediction vector of our CKT, deeper blue indicates better mastery of this question. The y-axis represents question id, and x-axis represents student's interactions with these questions.

For example, as shown in Figure 4(a), when a student has correctly answered question 60 continuously, the mastery level of question 60 becomes dark blue, that means this student has mastered question 60. In the beginning of Figure 4(b), the student has incorrectly answered questions 18 continuously, so the mastery level of question 18 becomes less and less, and finally turns white.

6.2 Discover Exercises Relationships

The second application of knowledge tracing is to discover the dependencies between different exercises which was used to be annotated by education experts. We follow the dependencies exploration method of DKT [23], and assign an influence J_{ij} to every directed pair of exercises i and j in training data,

$$J_{ij} = \frac{y(j|i)}{\sum_k y(j|k)} \tag{6}$$

where $y(j|i)$ is the correctness probability of question j from our model at the second timestep , given that a student answered exercise i correctly at the first timestep.

We perform above method on ASSIST2009 training dataset, and capture questions dependencies as shown in Figure 5. For better visualization, we ignored edges whose weight is less than 0.05. Although there is no ground truth about questions classes, some useful patterns can be found in the dependencies graph, we use Geiph to calculate the Modularity Measure [20] of this graph, and detect several communities marked as different colors, which represent some hidden concepts in ASSIST2009. The details of question id can be found in Figure 6.

7 CONCLUSION

In this paper, we proposed a novel model which is called *Convolution-augmented Knowledge Tracing* (CKT) for knowledge tracing problem. We use an LSTM to capture long-term features and use a 3D convolution network (3D ConvNets) to strengthen short-term features from recent student-question interactions. An update gate is proposed to adaptively fuse the two features and produce final prediction. Extensive experiments prove that our CKT model outperforms state-of-the-art models. In the future, we will consider proposing a higher-quality dataset based on our own education

platform, improving the existing classification KT problem into a regression problem, to facilitate more accurate modeling of knowledge states.

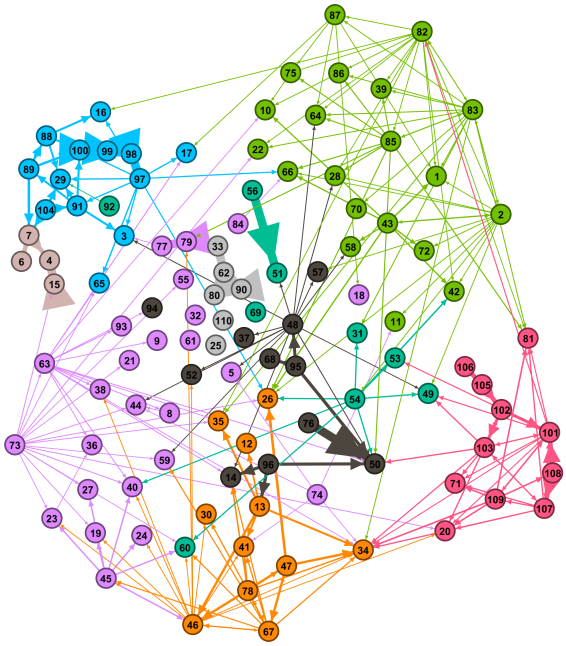

Figure 5: The dependencies of exercises from ASSIST2009 dataset.

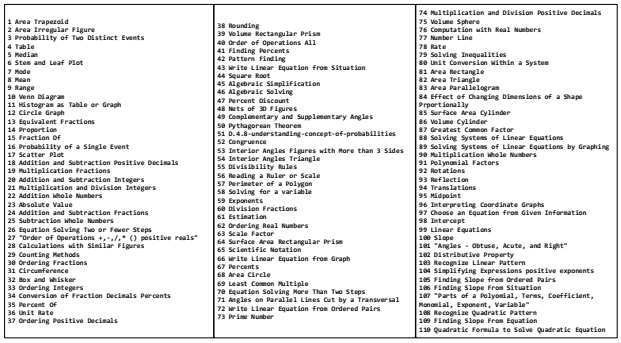

Figure 6: Question id corresponding with question name.

REFERENCES

[1] Ghodai Abdelrahman and Qing Wang. 2019. Knowledge Tracing with Sequential Key-Value Memory Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 175–184.

[2] Jiri Belohradsky, David Monge, Filip Zelezny, Matej Holec, and Carlos Garcia Garino. 2011. [IEEE 2011 24th International Symposium on Computer-Based Medical Systems (CBMS) - Bristol, United Kingdom (2011.06.27-2011.06.30)] 2011 24th International Symposium on Computer-Based Medical Systems (CBMS) - Template-based semi-automatic workflow construc. (2011), 1–6.

[3] Hao Cen, Kenneth Koedinger, and Brian Junker. 2006. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *International Conference on Intelligent Tutoring Systems*. Springer, 164–175.

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

[5] Min Chi, Kenneth R Koedinger, Geoffrey J Gordon, Pamela Jordon, and Kurt VanLahn. 2011. Instructional factors analysis: A cognitive model for multiple instructional interventions. (2011).

[6] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

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

[9] Hermann Ebbinghaus. 2013. Memory: A contribution to experimental psychology. *Annals of neurosciences* 20, 4 (2013), 155.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, 6645–6649.

[11] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. 2016. Hybrid computing using a neural network with dynamic external memory. *Nature* 538, 7626 (2016), 471–476.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 2961–2969.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Zhenya Huang, Yu Yin, Enhong Chen, Hui Xiong, Yu Su, Guoping Hu, et al. 2019. EKT: Exercise-aware Knowledge Tracing for Student Performance Prediction. *IEEE Transactions on Knowledge and Data Engineering* (2019).

[15] George J Klir and Bo Yuan. 1996. Fuzzy sets and fuzzy logic: theory and applications. *Possibility Theory versus Probab. Theory* 32, 2 (1996), 207–208.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[17] Sein Minn, Yi Yu, Michel C Desmarais, Feida Zhu, and Jill-Jênn Vie. 2018. Deep knowledge tracing and dynamic student classification for knowledge tracing. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1182–1187.

[18] Koki Nagatani, Qian Zhang, Masahiro Sato, Yan-Ying Chen, Francine Chen, and Tomoko Ohkuma. 2019. Augmenting Knowledge Tracing by Considering Forgetting Behavior. In *The World Wide Web Conference*. 3101–3107.

[19] Hiromi Nakagawa, Yusuke Iwasawa, and Yutaka Matsuo. 2019. Graph-based Knowledge Tracing: Modeling Student Proficiency Using Graph Neural Network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 156–163.

[20] Mark E. J. Newman and Michelle Girvan. 2004. Finding and Evaluating Community Structure in Networks. *Physical Review E Statistical Nonlinear and Soft Matter Physics* 69, 2 Pt 2 (2004), 026113.

[21] Shalini Pandey and George Karypis. 2019. A Self Attentive model for Knowledge Tracing. In *Proceedings of the 12th International Conference on Educational Data Mining, EDM 2019, Montréal, Canada, July 2-5, 2019*.

[22] Philip I Pavlik Jr, Hao Cen, and Kenneth R Koedinger. 2009. Performance Factors Analysis–A New Alternative to Knowledge Tracing. *Online Submission* (2009).

[23] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. In *Advances in neural information processing systems*. 505–513.

[24] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[25] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. 2016. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*. 1842–1850.

[26] Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence* 39, 11 (2016), 2298–2304.

[27] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[28] Yu Su, Qingwen Liu, Qi Liu, Zhenya Huang, Yu Yin, Enhong Chen, Chris Ding, Si Wei, and Guoping Hu. 2018. Exercise-enhanced sequential modeling for student performance prediction. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[30] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. 2015. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 4489–4497.

[31] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 6450–6459.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[33] Michael V Yudelson, Kenneth R Koedinger, and Geoffrey J Gordon. 2013. Individualized bayesian knowledge tracing models. In *International conference on artificial intelligence in education*. Springer, 171–180.

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