Expert Systems With Applications 196 (2022) 116670

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

GameDKT: Deep knowledge tracing in educational games

Danial Hooshyar [a], Yueh-Min Huang [b], Yeongwook Yang [c,*]

[a] School of Digital Technologies, Tallinn University, Tallinn, Estonia
[b] Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan
[c] Division of Computer Engineering, Hanshin University, Osan, South Korea

ARTICLE INFO

Keywords:
Learner model
Deep knowledge tracing
Educational game
Prediction of player performance
Deep learning

ABSTRACT

Despite the multiple deep knowledge tracing (DKT) methods developed for intelligent tutoring systems and online learning environments, there exists only a few applications of such methods in educational computer games. One key challenge is that a player may deploy several interweaved and overlapped skills during gameplay, making the assessment task nontrivial. In this research, we present a generalizable DKT approach called GameDKT that integrates state-of-the-art machine learning with domain knowledge to model the learners' knowledge state during gameplay, in an attempt to monitor and trace their proficiency level for the different skills required for educational games. Our findings reveal that GameDKT approach could successfully predict the performance of players in the coming game task using the cross-validated CNN model with accuracy and AUC of roughly 85% and 0.913, respectively, thus outperforming the MLP baseline model by up to 14%. When the performance of players is forecasted for up to four game tasks in advance, results show that the CNN model can achieve more than 70% accuracy. Interestingly, this model seems to be better and faster at identifying local patterns and it could achieve a higher performance compared to RNN and LSTM in both one-step and multi-step prediction of learners' performance in game tasks.

1. Introduction

Personalised learning refers to an educational approach that involves customisation of learning to suit the needs and interests of an individual. Multiple research reports the potential of personalised learning approach in improving learning outcomes and effectiveness as well as engagement and motivation of learners (Falcão, e Peres, de Morais, & da Silva Oliveira, 2018; Hooshyar, Yousefi, & Lim, 2018; Whitney, 2021). Identifying the learners' knowledge state (KS), which represents their proficiency level in a set of knowledge components (KC), is considered essential to achieve personalised learning (Huo, Wong, Ni, Chao, & Zhang, 2020). By accurately estimating the learners' KS, educators can identify the KCs learners are weak and strong at, and accordingly encourage them to work on it. Furthermore, such estimation enables an efficient diagnosis of an individual's strengths and weaknesses, which can later be used to suggest more appropriate learning materials, offer individualised feedback, recommend timely and personalised learning paths, and so forth (Corbett & Anderson, 1994; Piech, Spencer, Huang, Ganguli, Sahami, Guibas, & Sohl-Dickstein, 2015).

Knowledge tracing (KT) is considered a key approach when tackling this challenge (Piech et al., 2015). The KT method is considered as a family of learner modelling approaches that explores the students' past interactions with questions, items, and tasks (attempts on each KC), and accordingly predicts their future performance (Abyaa, Idrissi, & Bennani, 2019; Hooshyar et al., 2020). In other words, by observing the past interaction of learners and KCs, KT approach infers the learners' KS (latent). Bayesian KT (BKT) and deep KT (DKT) are among the most important methods in KT (Corbett & Anderson, 1994; Pelánek, 2017). From an architectural perspective, these two methods are different in the sense that DKT benefits from deep learning frameworks. BKT, which is considered a more traditional method of KT, models the discrete state of learners' mastery of a specific KC. However, the DKT approach often employs some variant of multi-layer neural networks that can effectively deal with sequential data, for instance long short-term memory units (LSTM) or recurrent units (RNN). In RNNs, for instance, the data points in a sequence are recursively processed, which allows to retain information gained during steps t_0 to t_{n-1} for predicting t_n. Multiple researchers claim that learning through exercises and answering questions could be considered a type of sequential task (Piech et al., 2015; Wang, Sy, Liu, & Piech, 2017a).

* Corresponding author.
E-mail addresses: danial.hooshyar@gmail.com (D. Hooshyar), huang@mail.ncku.edu.tw (Y.-M. Huang), yeongwook.yang@gmail.com (Y. Yang).

https://doi.org/10.1016/j.eswa.2022.116670
Received 6 September 2021; Received in revised form 27 December 2021; Accepted 8 February 2022
Available online 14 February 2022

Although there exist numerous works that focus on (deep) KT in various online learning environments, there is only a few research that deals with the problem of modelling the players' knowledge through DKT in the context of educational games (e.g., Kantharaju et al., 2018). Essentially, this implies estimating the player's knowledge or mastery level in a specific set of skills or concepts. One key challenge in applying DKT to educational games is that, unlike intelligent tutoring systems, it is difficult to define whether a student was successful in applying the specific skills, and the instance when a student applies a specific skill. One assumption in (deep) KT is that students can successfully or unsuccessfully apply specific skills in a straightforward, easy manner. In educational games, nonetheless, a student may deploy several overlapped and interweaved skills during gameplay, making the assessment task nontrivial. For instance, to solve a task, students may drag and drop different elements to merely explore their behaviour within the game. This makes it difficult to evaluate whether students applied a specific skill correctly.

Inspired by the existing works on student modelling that employs DKT, we propose a new DKT approach (called GameDKT) that models the learners' knowledge state during gameplay, to monitor and trace their proficiency level for the different skills needed to play the educational game. GameDKT approach is integrated in an educational game that teaches computational thinking called AutoThinking (Hooshyar et al., 2019). The game offers adaptivity in both gameplay and learning process by customising the behaviour of non-player characters as well as feedback and tutorials suited to players' needs and level of knowledge. To provide this adaptation, the game requires continuous estimation of players' level of proficiency and knowledge. In particular, GameDKT approach tackles the above-mentioned challenge by combining machine learning with domain knowledge to make predictions with regard to successful application of skills by identifying the applied skills and algorithmic thinking strategy currently employed by the students, using a series of time windows. The reason is that in some situations, not only is it hard to specifically define the instance when students succeed or fail to deploy skills in general, but also it is difficult to infer whether the students' application of skills are a sign of their mastery or chance. For instance, a player might employ an unsuccessful algorithmic thinking strategy, leading to successful or unsuccessful application of some part of the essential skills. Therefore, we also consider the algorithmic thinking strategy that is used to apply those sets of essential skills/concepts. We assume two algorithmic thinking strategies of *random* and *parallel thinking,* where the former refers to solving the current game task by repeatedly trying different solutions without any specific purpose, and the latter indicates the process of considering and purposefully applying multiple skills simultaneously. An example of *random thinking* strategy is employing high-level solutions in situations where an empty path is unnecessarily traversed over and over. An example of *parallel thinking* strategy is employing high-level skills to purposefully travel a (long) path aimed at collecting as much score as possible and escaping from the non-player characters (NPCs). It is worth mentioning that not all *random* strategies end up with losing scores, some may even lead to gaining scores by chance.

Similar to the approach proposed by Kantharaju et al. (2018), our assumption is that while players employing *parallel thinking* implies successful application of the essential skills in the game task, them deploying *random* strategy implies that learners do not have the required skill proficiency or do not fully understand the logic behind the game level or task. Predicting or considering the algorithmic thinking strategy can be considered as a way to understand whether those learners have successfully understood and applied the required skills involved in a certain game task. This information is then fed to the GameDKT approach. Note that even though more algorithmic thinking strategies can be identified, this research only considers *random* and *parallel thinking*. The following are the contributions of our work:

- Proposing and validating a new approach that only requires game task-solution logs of each player and can be applied on any players' input that can be vectorised for modelling player knowledge, and accordingly predict their future performance in educational games.
- Timely tracing of learners' knowledge state in an educational game for teaching computational thinking.
- Benefiting from state-of-the-art (sequential) prediction models such as recurrent neural networks, long short-term memory, and convolutional neural networks for learning hidden representations of players' actions and concepts underlying the game tasks, and accordingly trace their knowledge state in educational games.
- Employing and measuring the effect of sequence padding and replacing missing sequential values on the performance of deep learning-based models.
- Investigating the effect of different sequence lengths of actions and dataset types (original, padded, and replaced missing values) on prediction power of different deep learning-based models.
- Comparing the performance of models on one-step prediction versus multi-step prediction of learners' performance in game tasks.

2. Related work

2.1. Knowledge tracing

The main aim of knowledge tracing is to use the learners' past learning performance to in turn, learn about their knowledge representation over time. Usually, predicting learners' performance on future tasks or questions is considered as a way to evaluate knowledge representation. Two of the most well-known approaches of knowledge tracing are Bayesian knowledge tracing (BKT) and deep knowledge tracing (DKT). The former approach relies on hidden Markov models to the model prior knowledge and speed of learners (Yudelson, Koedinger, & Gordon, 2013). This approach takes into account whether a learner fails to answer a question in spite of mastering the skill (called slipping), and whether a learner guesses the answer to a question without mastering the skill (called guessing), in an attempt to model latent information in a learner's stochastic learning process. Past data is then fed to expectation–maximisation algorithms to optimise these parameters and those of the hidden Markov models (Pelánek, 2017). According to these estimated parameters, the BKT predicts future performance of learners. Several researchers proposed different variants of the original BKT approach; for instance, Khajah, Wing, Lindsey, and Mozer (2014) integrated latent elements such as individual differences to improve model performances; Heathcote, Brown, and Mewhort (2000) proposed listing the learners' performance sequence repeatedly to link the performance on multiple skill questions or tasks with all essential skills, making the model observe this piece of evidence several times for each one of the required skills; Khajah, Lindsey, and Mozer (2016) took into account the fact of forgetting the skills by incorporating a parameter called forgetting. While BKT approaches have the advantage of offering a good interpretability, they have some weaknesses including an inability to handle multi-skills simultaneously and requiring manual (topic) labelling.

On the other hand, DKT approaches that deploy sequence to sequence models to represent knowledge states has shown to achieve more promising results compared to BKT approaches on various domains including learning language (Osika, Nilsson, Sydorchuk, Sahin, & Huss, 2018), engineering statics (Zhang, Shi, King, & Yeung, 2017), programming exercises (Wang et al., 2017a), and mathematics (Xiong, Zhao, Van Inwegen, & Beck, 2016). Much like BKT, researchers have proposed multiple variants of DKT approaches as well. For instance, Piech et al. (2015) modelled students' learning by applying DKT using RNN, and found that their proposed approach outperforms other existing methods. Zhang et al. (2017) proposed a knowledge tracing approach that stores concepts of learners' understanding and knowledge in two static matrices in order to discover some underlying concepts. In

another attempt, Su et al. (2018) employed one RNN to encode learner's interactions with questions, and another RNN to encode questions text to provide a better characterisation of connection between questions. Yeung (2019) combined deep learning-based methods (e.g., RNN) with item response theory to provide interpretability for the estimated skills. Kim, Kim, Jung, and Kim (2021) introduced a new method called DiKT for knowledge tracing network with a dichotomous perspective on a student's knowledge state. Liu et al. (2021) introduced an approach that simulates human memory in knowledge tracing. This approach, called Hierarchical Memory Network or HMN, uses an external memory matrix and two mechanisms of divide and decay to achieve a hierarchical memory. Results of their experiments revealed that their proposed HMN approach outperformed other existing models, had a strong ability to generalize, and could learn faster.

Despite the diversity among the proposed DKT approaches, their effectiveness in educational computer games has not been evaluated. They mostly ignore evaluating the performance of other variants of deep learning-based models such as convolutional networks. These approaches mostly use a non-stationary sequence length of learners' actions and do not assess the effect of different sequence length sizes on the performance of their model. They overlook the fact about investigating the effect of replacing missing values in their time series (or sequential) data on the performance of their proposed approach, and often ignore comparing this performance in one-step prediction versus multi-step prediction of learners' performance.

2.2. Deep knowledge tracing in educational computer games

There are a few research that studies the application of knowledge tracing in educational computer games. For example, Gweon et al. (2015) proposed a BKT model for educational games that – with the development of a Monte Carlo model – identifies a source of difficulties, and accordingly traces learners' online learning activities. Schodde, Bergmann, and Kopp (2017) presented a BKT-based approach for personalising language tutoring in a tutoring interaction that resembles a game. The approach selects the next action for learners as it can keep track of their knowledge state. Authors reported promising results from their evaluation in the sense that students using the proposed approach appeared to have a higher learning performance. Moreover, Cui, Chu, and Chen (2019) presented a BKT approach that compares Bayesian models with dynamic Bayesian models in tracing the learners' skill in an educational game called Raging Skies. Results of their experiment reveal the superiority of Bayesian knowledge tracing compared to dynamic Bayesian knowledge tracing. One challenge of using the aforementioned approaches is that not only are they unable to allow more than one skill or piece of knowledge at each game level, but also, they sometimes require a certain number of data points to fit the parameters of BKT model.

To address such challenges, Kantharaju et al. (2018) introduced a knowledge tracing approach that utilises machine learning and expert knowledge to predict if a player deploys the required skills successfully or unsuccessfully. Authors evaluated the proposed approach using data from an educational game called Parallel and reported that their approach can predict the students' understanding of skills with a low mean-squared error. While this approach may successfully overcome some of the challenges of the previously proposed BKT-based approaches, it certainly overlooks the power of various deep neural network-based models, such as RNN, LSTM, or CNN, that can effectively deal with sequential data in modelling latent information of learners, which is considered as a type of sequential task. Needless to say, it neither explores the use of padding or replacing missing values in the sequence of actions in datasets, nor attempts to compare the performance of their proposed approach in one-step prediction versus multi-step prediction of learners' performance. To build on the existing approaches of knowledge tracing in educational game, this research proposes the first DKT approach called GameDKT that tackles the issues of previous approaches; it only requires game task-solution logs of each player and can be applied on any educational games in which the players' input that can be vectorised.

3. GameDKT: Deep tracing of Players' knowledge in educational games

In this research, we propose an approach, GameDKT, for predicting successful/unsuccessful applications of skills from a series of time windows in an educational game. In the following sections, we firstly formulate the deep knowledge tracing problem, then describe the proposed GameDKT approach, and finally introduce the models used in the approach.

3.1. Formulation of deep knowledge tracing problem

To play and complete a level on the AutoThinking game, players can develop up to 20 solutions in sequence. Simply put, the players, whose character is that of a mice, should develop solutions that allow them to collect scores (by eating cheese pieces) and escaping from NPCs (two cats). During the game, players can employ low- or high-level solutions that involve sequences, patterns and functions, loop and conditionals, and so forth. For each student, we focus on the learners' sequence of solution submissions and their success in level three of the game. In observing the students' sequence of solution submission attempts over time on a CT game task, our task is to predict whether the student will successfully complete the next CT tasks within the same game episode.

According to the deep knowledge tracing definition adopted, a correct skills/knowledge label indicating whether the student correctly applied the required skills using a proper algorithmic thinking strategy is generated for each student at each time step (i.e., for each solution throughout the game). While doing so, we also considered whether a player's algorithmic thinking strategy (while applying the skills) was *random* or *parallel*. The reason for this is that in some situation, not only is it hard to specifically define when students succeed or fail to deploy skills in general, but it is also difficult to infer whether the students' application of skills is a sign of their mastery or chance.

Suppose a game level spans over 20 solutions as depicted in Fig. 1. For a specific student, we obtain her task-by-task solutions activities in the game, denoted by an input sequence $(x_1, x_2, \cdots, x_t)$, and the corresponding skills/knowledge labels forming an output sequence $(y_1, y_2, \cdots, y_t)$. As the game progresses, our goal is to predict the skills/knowledge label of a *k* student task through *k-1* previous solutions. We view the deep knowledge tracing task as a sequence labelling or classification problem, since the activities are recorded in a task-by-task format, an input sequence and the corresponding skills/knowledge labels assigned according to the chosen knowledge tracing definition form an output sequence.

To tackle this challenge, we employ the idea of "time window" to facilitate the identification of successful/unsuccessful skill application attempts in each time window. More explicitly, we create fixed time windows of game data for each player. In other words, considering a sequence of learners' actions and a tuneable time interval size τ, we extract a feature vector based on the information on learners' solutions and actions from each time interval (t, t + τ) at intervals of $\tau/2$. From each window, a collection of different actions and solutions that occurred in each game is computed. This includes player ID, type of the game task, skills required to solve the tasks, time differences between certain actions, and whether the player has successfully applied the required skill(s)/strategy to solve the specific task. With regard to the detection of additional evidence of successful or failed skill application, a set of defined domain knowledge rules are applied to each time window. For example, where there are more than 3–4 titles and/or junctions ahead of the mouse, skills of sequence, loop, and/or conditional are necessary, as well as *parallel thinking* strategy. Needless to mention that according to the position of the NPCs in the game, pattern detection and

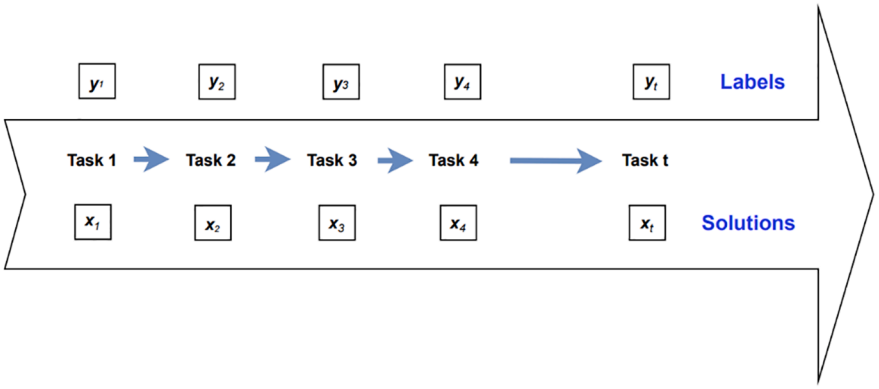


Fig. 1. Formulation of knowledge tracing problem.

generalization skill could also be applied. Once the required skills are correctly applied using a proper algorithmic thinking strategy, we signal successful skill application. Using these features, a knowledge model of a player will be built, which is aimed at sequential tracing of their knowledge.

3.2. The proposed approach

Fig. 2 illustrates the general overview of GameDKT approach. As shown below, in the *first phase*, the students' sequence of solution submissions is collected and pre-processed. For this purpose, a combination of task types and algorithmic thinking strategy, whether these two are properly applied as well as learners' ID are considered. To be more specific, since some players developed less than the standard 20 solutions required to complete the game, which may or may not have resulted in them winning the game, we employed two techniques to create a stationary submission set for each player. These include padding and replacing missing values using linear interpolation between the two neighbouring values in the series, which calculates the replacement value. Thereafter, a correct skill/knowledge label that indicates whether the student correctly applied the required skills using a proper algorithmic thinking strategy is generated for each student. We observe that,

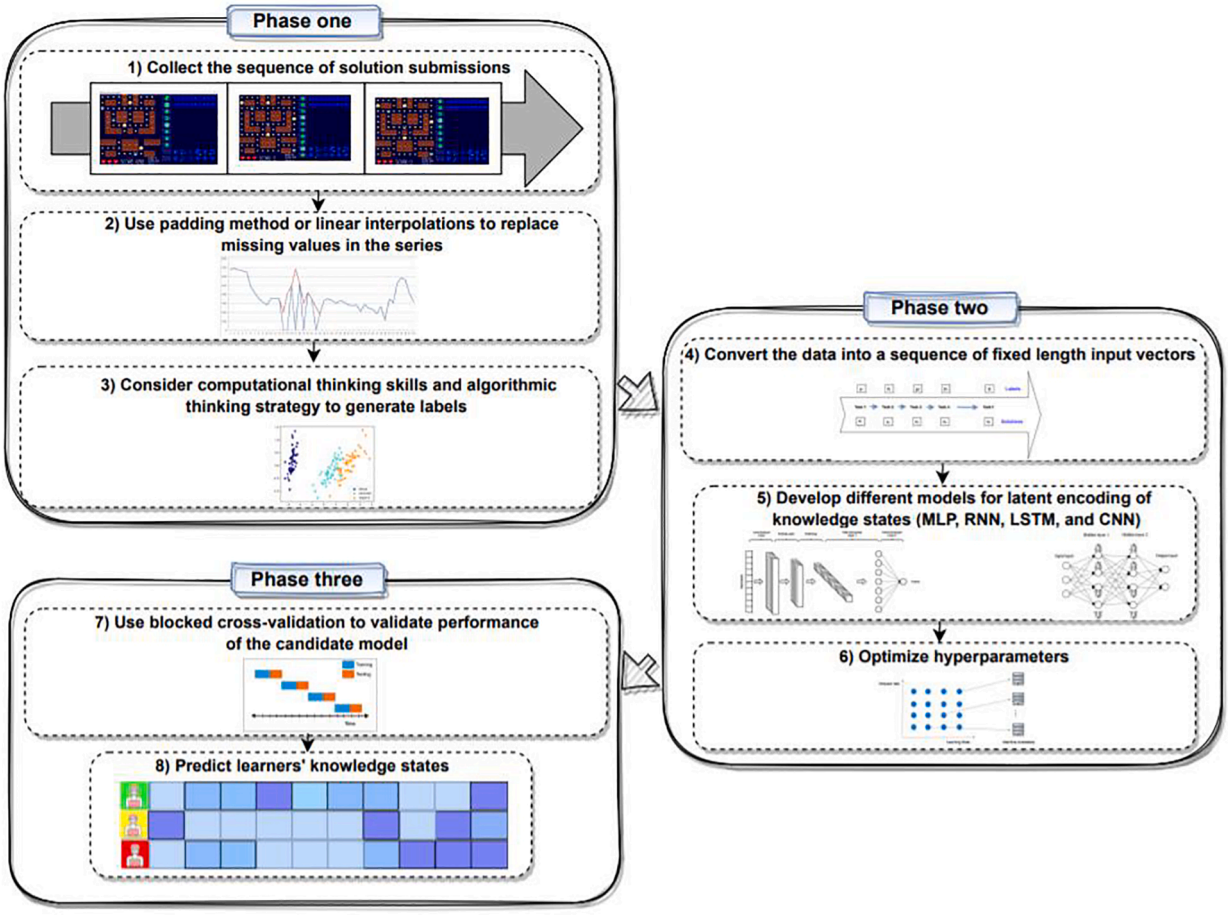


Fig. 2. General overview of GameDKT approach.

unlike traditional approaches of knowledge tracing, we did not manually label the association between the learners' interaction and their corresponding skills or algorithmic thinking strategy, and various types of deep neural network-based models were designed and employed to take vectorised inputs. These properties make the model an appropriate fit to understand the trajectories of open-ended student responses, which have unbounded input spaces (Hinton & Salakhutdinov, 2006).

In the *second phase*, we converted our sequential data into a sequence of fixed length input vectors of 10, 15, and 19. Furthermore, we employed a range transformation method to normalise our data to a specified value range. We used this method because our dataset did not contain outliers and this method keeps the original distribution of the data points. Due to a rather low-dimensional space, unlike other research in the area (e.g., Wang, Feng, Tang, Huang, & Liu, 2019), we did not use feature embedding. After generating a sequence of fixed length input vectors, four state-of-the-art classification models, namely convolutional neural networks (CNN), long short-term memory units (LSTM), recurrent units (RNN), and multilayer perceptron (MLP) are designed and employed to encode the latent knowledge states of learners (see the following section for more details). Network architectures of the models are varied with different numbers of state dimensions and memory size. An evolutionary optimisation method, i.e., Genetic algorithm, was applied over the combinations of state dimensions and memory size. As for the LSTM model, we used a batch normalisation layer, a LSTM layer with seven neurons (with activation function of ReLU), and a softmax layer with two neurons. Similarly, the RNN-based approach used the same hyperparameters as LSTM, but by replacing the LSTM layer with a recurrent layer. The CNN model, however, utilised deep learning using a convolutional layer with an activation map of 64, ReLU activation function, a kernel size of three, a fully connected layer with 50 neurons, and a softmax with two neurons. Lastly, the MLP model (baseline) includes three fully connected layers with 50 neurons each.

Finally, in the *third phase*, in order to validate the performance of our models, a blocked cross-validation with a training and testing window size of 30 were used with independent test sets (see Section 4.2 for more details). To show the accuracy, stability, and robustness of the designed model, performance of the models was tested not only to predict one step ahead of the sequence (performance of learners in the next game task), but also to predict up to five steps ahead of the action sequences.

3.3. Models

The GameDKT approach benefits from several state-of-the-art machine learning models to learn hidden representations of players' actions. These include RNN, CNN, LSTM, and MLP, which is our baseline.

3.3.1. MLP

A multilayer perceptron (MLP) is one of the first feed-forward artificial neural networks (Svozil, Kvasnicka, & Pospichal, 1997). It consists of three layers, namely input, hidden, and an output layer that are connected to each other. Each layer consists of neurons from one to many. And each neuron consists of three main components, namely input x, output y, and bias b. It uses the following non-linear activation function:

$$\mathrm{y_i} = \sigma(\sum_i w_i x_i + b_i)$$

y is the output calculated by a weighted linear summation as $w\bullet$x, and σ is an activation function that can be tanh (hyperbolic tangent function), sigmoid, or rectified linear unit function for gradient-decent-based learning. MLP uses the backpropagation technique that is a supervised learning technique for training (Hecht-Nielsen, 1992).

3.3.2. RNN

Recurrent neural network (RNN) is one of the architectures for artificial neural networks to learn sequential data set (Zaremba, Sutskever, & Vinyals, 2014). It has three types of layers, namely input layer *w*, recurrent layer *r*, and output layer *y*. The activation of each layer at time *t* is denoted as *w(t)*, *r(t)*, and *y(t)*, respectively. RNN calculation can be presented at time *t* in the following manner:

$$x(t) = [w(t)r(t-1)]$$

$$r(t) = f_1(U\bullet \mathrm{x(t)})$$

$$y(t) = g_1(V\bullet r(t))$$

where *x(t)* is a vector that concatenates *w(t)* and *r(t-1)*, *f1(.)* and g1(.) are element-wise sigmoid and softmax function, respectively, and *U* and *V* are weights which will be learned.

3.3.3. LSTM

Long short-time memory (LSTM) is a recurrent neural network architecture for deep learning (Hochreiter & Schmidhuber, 1997). It has three gates, namely input, forget, and output gate. Each gate uses a sigmoid function for input and hidden state. LSTM calculation is as follow:

$$\mathrm{i_t} = \sigma\left(x_t U^i + h_{t-1} W^i + b_i\right)$$

$$\mathrm{f_t} = \sigma\left(x_t U^f + h_{t-1} W^f + b_f\right)$$

$$o_t = \sigma(x_t U^o + h_{t-1} W^o + b_o)$$

$$q_t = \tanh\left(x_t U^q + h_{t-1} W^q + b_q\right)$$

$$p_t = f_t\bullet \mathrm{p_{t-1}} + \mathrm{i_t}\bullet \mathrm{q_t}$$

$$\mathrm{h_t} = o_t\bullet \tanh(\mathrm{p_t})$$

where i_t denotes input gate, f_t is forget gate, and o_t is output gate. And x_t is an input state while $\mathrm{h_{t-1}}$ is hidden state. To generate the hidden state at current step *t*, temporary result q_t is first generated by the ensemble of input x_t and tanh (hyperbolic tangent function) nonlinearity with respect to the previous hidden state $\mathrm{h_{t-1}}$. Hence, temporary result q_t and history p_{t-1} are combined with input gate i_t and forget gate f_t, respectively, to get an updated history p_t. For the last updated history p_t, we use the output gate o_t to get the final hidden state h_t.

3.3.4. CNN

Convolutional neural network (CNN) is one of the most popular architectures for deep learning and it's used for extracting the dominant information on a sequence data (Albawi, Mohammed, & Al-Zawi, 2017). It usually consists of input, convolution, and pooling layer. Input layer presents a sequence that contains entries. Each entry is represented by a d-dimensional dense vector. For example, given a sequence x and n entries are represented as a feature map of dimensionality $d \times n$.

Convolution layer is used for learning features that are sliding w-grams. The following is the equation for convolutional weights:

$$p_i = \tanh(\boldsymbol{W}\bullet \mathrm{c_i} + \mathrm{b})$$

where $\boldsymbol{W}$ is convolution weights as $\boldsymbol{W} \in \mathbb{R}^{d\times wd}$ and vector c_i is the concatenated embedding of w entries. Also, b is bias as $b \in \mathbb{R}^{\mathrm{d}}$.

Pooling layer is used to generate the representation of input sequence x by using all w-gram representations p_i and max pooling. The following is the equation for max pooling:

$$x_j = \max(p_{i,j}, p_{2,j}, \cdots)(j = 1, \cdots, d)$$

4. Experimental evaluation

4.1. Datasets

Datasets used in this research comprises log data of learners playing the third level of AutoThinking game in Estonia, Taiwan, France, South Africa, and South Korea. Learners were selected from different age groups and backgrounds. The dataset has 5981 examples/solutions from 408 learners who had played the game between December 2019 and August 2021. Some learners played the game in the context of experimental studies (e.g., Hooshyar et al., 2021, El Mawas, Hooshyar, & Yang, 2020), while others played on their own. As mentioned, some learners had used 20 solutions to complete the game, whereas others used less than 20.

Off all the attributes collected from the learners – current position of the mouse and NPCs; big and small cheeses, walls, and junctions; number of times small and big cheeses are cleared using each solution; number of times the loop, conditional, functions, and arrows are used; number of times debug, simulation, or help was used; number of given textual, graphical, or video feedback; number of lives left; and number of times a solution led to hitting walls – in this research, we considered the following: learners' ID, types of game tasks or algorithmic thinking strategy, skills or strategy required to solve the task, and whether the learner correctly applied the required skills/strategy to the corresponding task. However, to generate some of these features, we considered multiple attributes. For instance, in order to label whether the employed skills or strategy was successful, we considered the following: number of times the mouse hits the walls using the current solution, score for the specific solution (including the scores for eating cheeses as well as skills), the skill or algorithmic thinking strategy that was employed in the developed solution, distance of the NPCs to the mouse, and so forth. To specify or separate the game tasks, we took into account the distance of the mouse from both NPCs, current position of the mouse, NPCs as well as big, fixed and moving cheeses, junctions, and walls, and so forth.

We test the ability to predict student performance on multiple datasets, namely original AutoThinking (OA), padded AutoThinking (PA), and replaced missing values AutoThinking (RMA) datasets. Additionally, we produced three different versions of each dataset with a sequence length of 10, 15, and 20. While all the three datasets use data from all 408 learners, the OA includes 5981 examples, and the PA and RMA datasets include 8160 solutions/examples. To create the PA dataset, we padded with zero to fill up the remaining time steps, whereas for the RMA dataset, we employed a linear interpolation between the two neighbouring values in the series to calculate the replacement value. Table 1 presents some samples from the PA and RMA datasets.

4.2. Experiment setting and evaluation

We implement a LSTM-, RNN-, and CNN-based, and a fully connected MLP model on a computer having a single Intel(R) Core™ i7-8650U CPU and 16.0 GB memory. We learn the models using ADAM optimisation with a learning rate of 0.03, 20 epochs, cross-entropy loss, using Gaussian distribution with zero mean and a small standard deviation, epsilon 1.0E-6, using backpropagation. In addition to this, since some of the input sequences are of different lengths, all sequences are set to be a length of 10, 15, and 19. For sequences that have less than 19-time steps, they are padded with zeros to fill up the remaining time steps. Accordingly, masking is also applied when computing the loss.

Table 1
Examples from PA and RMA datasets.

Index	User_ID	Game_Task	Response
0	0	6	1
1	0	2	1
2	0	7	1
3	0	1	0
…..	…..	…..	…..
8157	407	2	0
8158	407	4	0
8159	407	6	0

For all datasets, we evaluate our results using time-based cross-validation (blocked cross-validation with a training and testing window size of 30, with independent test sets), and in all cases, hyperparameters are learned on training data (see Fig. 3). The reason for employing blocked cross-validation is to mainly prevent leakage from future data to the model. Basically, this method creates two margins between the training and validation folds, and the folds used at each iteration. The former aims to prevent the model from observing lag values that are used twice, once as a regressor and another as a response (i.e., the test window step size), while the latter seeks to avoid the model from remembering patterns from an iteration to the next (i.e., the training window step size).

In brief, we used *k* fold to *k* instances to train a model that predicts "one-step" or "multiple steps" ahead. We set the training window size in such a way so as to ensure that each of our *k* instances uses only information available at that particular time. This assures that our out-of-sample test has not been polluted by accidentally peeking ahead into the future. Moreover, this helps to ensure that we do not accidentally pollute future information if some of our independent variables are asset returns (example of such pollution is training on the first 30 solutions and testing on a solution within the first 30). Thus, after validating our model, our proposed model forecasts the actual future. We compare the results of our proposed CNN-, RNN-, and LSTM-based approach to a standard MLP. Besides, we compare the results from our proposed approach in predicting the performance of students' one-step ahead with multiple steps ahead (namely, predicting performance of students in 2, 3, 4, and 5 future game tasks).

5. Results and analysis

In this section, performance prediction of each model is presented through two measures, namely area under curve (AUC) and square of Pearson correlation (R^2). Additionally, we used AUC values to create receiver operating characteristic curve (ROC). These measures offer robust information for evaluating the prediction models, where R^2 denotes the square of Pearson correlation coefficient between the observed and predicted values of dependent variable. Combining both measures account for different aspects of a model and provide a solid basis for evaluating it. Additionally, we present accuracy, precision, and recall for each model.

5.1. Performance of models on the OA dataset

Fig. 4 illustrates ROC measures of the cross-validated model predictions on the original AutoThinking dataset for different sequence

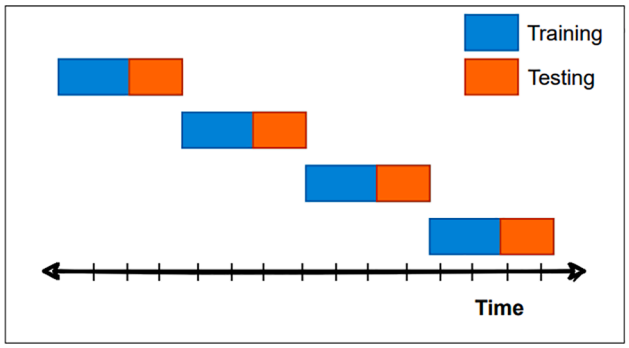
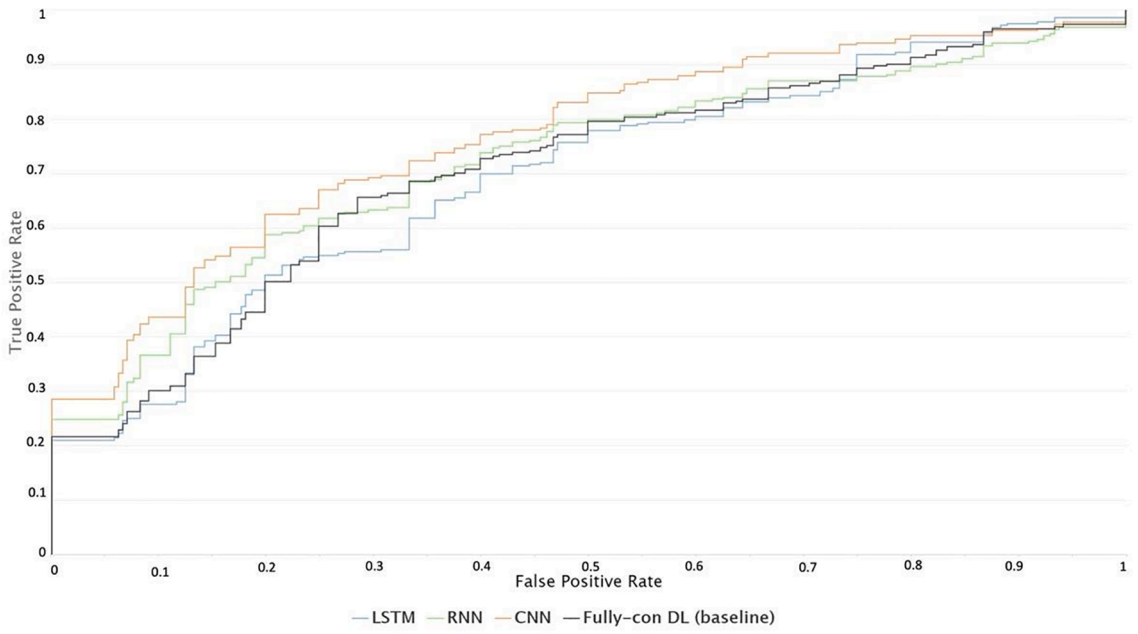
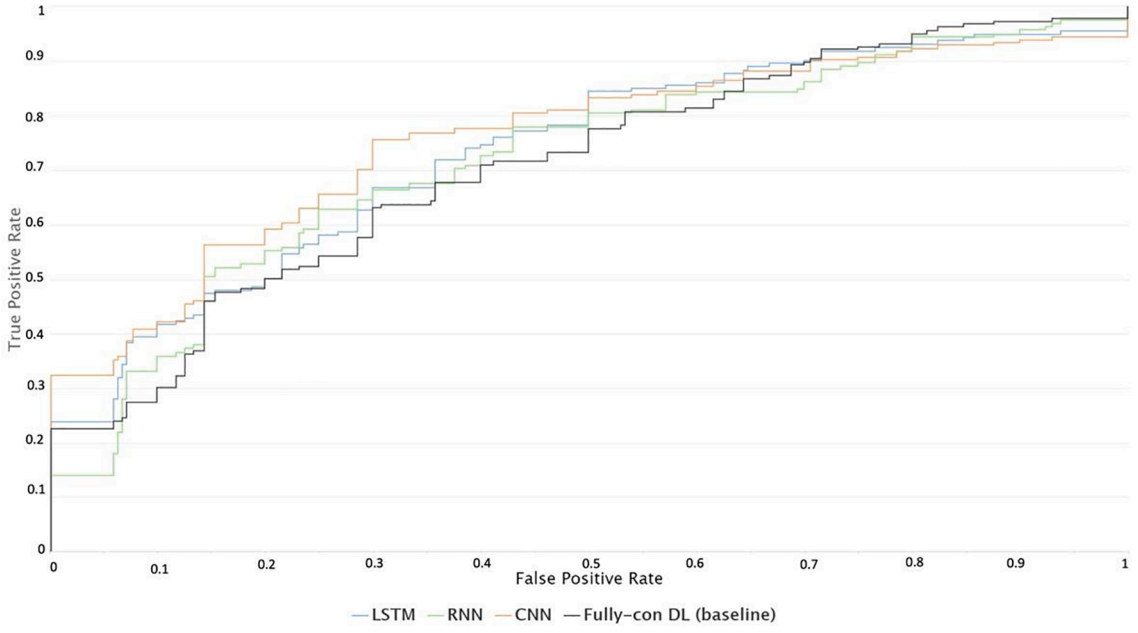


Fig. 3. Blocked cross-validation for sequential data.



a)

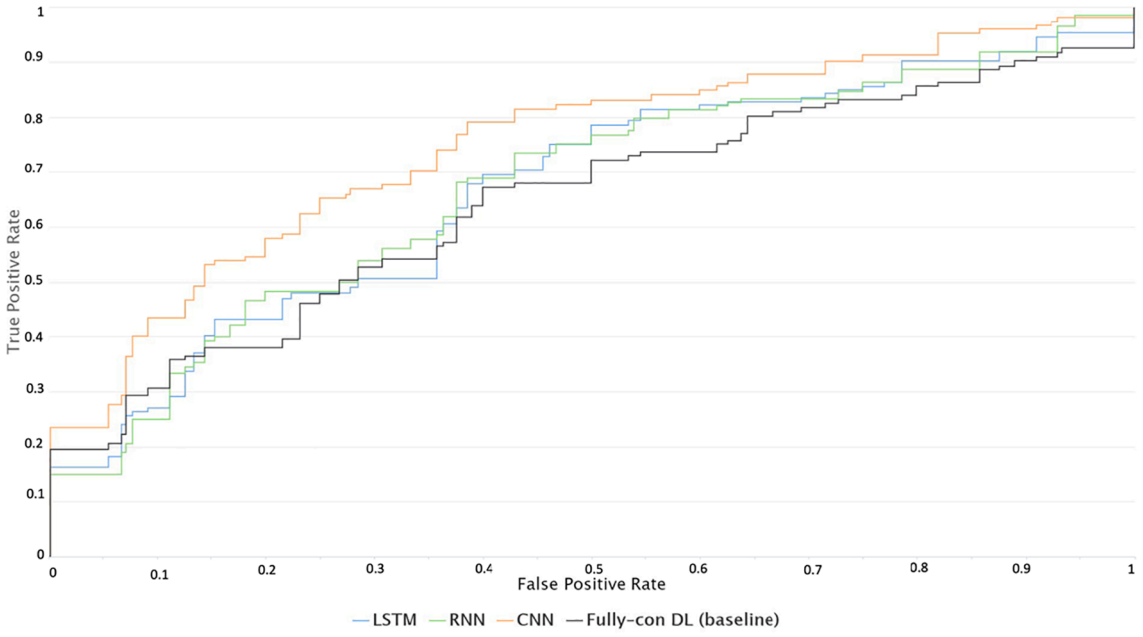


b)

Fig. 4. ROCs of classifiers over: a) 10, b) 15, and c) 19 sequence lengths of actions in original dataset.

lengths, while Table 2 presents AUC and squared correlation measures of the cross-validated model predictions. Furthermore, Fig. 5 presents accuracy, precision, and recall measures of these models for different sequence lengths. Overall, it is apparent that the largest sequence lengths produce rather lower results, while a sequence length of 15 archives the highest results. The highest and lowest performances were achieved by the CNN and baseline model on sequence lengths 15 and 19, respectively.

To be more specific, on a sequence length of 10, the CNN model outperformed other models, with an accuracy of 68.43, recall measure of 54.67, and precision of 64.64; the AUC and R^2 scores in this instance is 0.755 and 0.162, respectively. With regard to the sequence length of 15, CNN followed by LSTM model resulted in the highest performance with an accuracy of 69.39 and 68.18, respectively. These models also obtained the highest scores for AUC and R^2, which was 0.765 and 0.193, respectively, for CNN and 0.732 and 0.167, respectively, for LSTM. On a sequence length of 19, similarly, the CNN and LSTM models obtained the highest accuracy with 70.42 for CNN and 66.25 for LSTM, where the CNN model obtained an AUC score of 0.749 and R^2 score of 0.194, while the baseline model appears to have the lowest performance (i.e.,



c)

Fig. 4. (*continued*).

Table 2
AUC and squared correlation for classifiers over three sequence lengths for original dataset.

Sequence Length	LSTM		RNN		CNN		MLP (baseline)	
	AUC	R^2	*AUC*	R^2	*AUC*	R^2	*AUC*	R^2
10	0.693	0.086	0.722	0.128	**0.755**	**0.162**	0.705	0.140
15	0.732	0.167	0.718	0.138	**0.765**	**0.193**	0.708	0.121
19	0.667	0.173	0.672	0.133	**0.749**	**0.194**	0.644	0.060

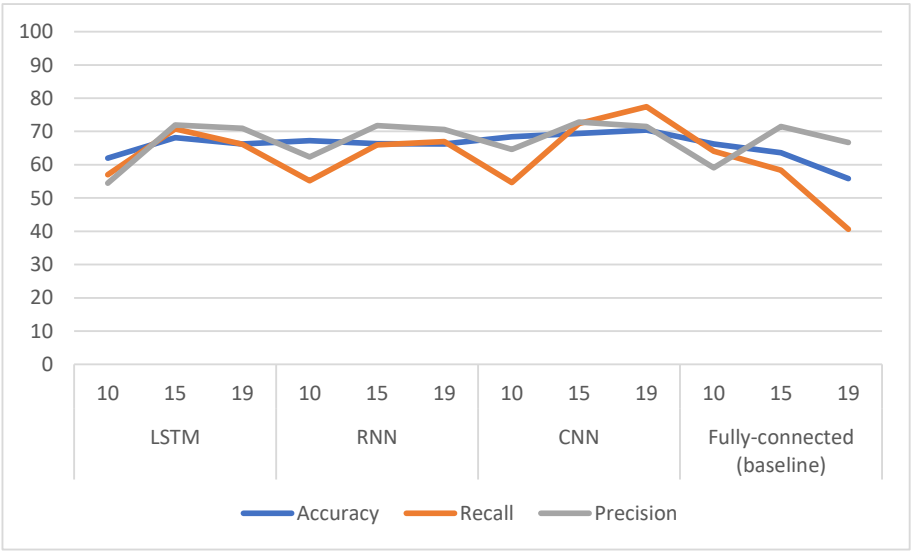


Fig. 5. Accuracy, precision, and recall measures of the models for different sequence lengths on the original dataset.

accuracy of 55.83). Obviously, with sequence length of 10, 15, and 19, the CNN model could produce around 5, 6, and 11 percent gain, respectively, over the MLP (baseline) model, constantly outperforming the state-of-the-art baseline by around 5% (see Table 2). Observe that as this is a 2-way classification problem; the baseline classification accuracy of a random classifier would be 50%, which indicates the efficiency and inefficiency of the CNN and MLP (baseline) models for this problem, respectively.

While analysing these models, we found that the LSTM model appeared to have the lowest performance among all models with regard to sequence length of 10. However, it achieved a performance that was roughly as good as the CNN model when increasing sequence lengths. The reason for this is that LSTM model keeps, in its memory, some single observations for a long time, which might not necessarily be useful, and one way to improve the performance of LSTM models is to employ larger or longer models that can remember (or learn) longer-term sequential dependencies. On the other hand, the RNN model seems to be amongst the best models in shorter sequence length (i.e., 10), and its performance decreased only slightly with larger sequence lengths. The reason for the high performance on rather shorter sequence is that it usually only looks at a few latest inputs (short-term modelling). Finally, the reason for high performance of the CNN model is that, compared to other models such as RNN and LSTM, it does not develop relationships between hidden vectors of each sequence step and can identify local patterns better. CNNs do not have the assumption that history is complete, and unlike other models, it learns patterns within the time window. Finally, performance of the baseline model seems to have decreased with an increment in sequence length, and it generally could not handle the sequential data well.

Consequently, although results from the best model on different sequence lengths could not exceed AUC and R^2 scores of more than 0.765 and 0.194, respectively, it seems logical to conclude that the results are fairly good given that different sequence lengths often use data points from more than one game episode to predict the performance of players on the upcoming game tasks. For instance, on a sequence length of 19, there might be 12 data points from player A and 7 from player B, to predict the performance of the eight tasks for player B, which could include or exclude data points from players on some specific game tasks. In the following section, it can be seen that often, models perform better on padded datasets or one where missing values have been replaced, where a stationary sequence of data points from one game/player is used to predict the performance of the same player on the next task in the same game episode.

5.2. Performance of models on the PA dataset

Table 3 illustrates AUCs and squared correlation measures of the cross-validated model predictions on the padded dataset for different sequence lengths, while Fig. 6 shows ROCs of the models on the padded dataset. Furthermore, Fig. 7 presents accuracy, precision, and recall measures of these models for different sequence lengths. Much like the original dataset, it is apparent that larger sequence lengths produce rather lower results, while sequence length of 15 archives the highest results. The highest and lowest performances were achieved by CNN and baseline models on sequence lengths 15 and 19, respectively.

More specifically, on a sequence length of 10, the CNN model outperformed other models, with a prediction accuracy of 74.78%, and an AUC score of 0.787. With regard to sequence length of 15, the CNN followed by RNN model resulted in the highest performance, with an accuracy of 84.17 and 83.12, respectively. These models also obtained the highest AUC and R^2 scores. On a sequence length of 19, LSTM and CNN models obtained the highest accuracy with 73.64 for LSTM and 72.73 for CNN, while the baseline model appears to have the lowest performance (i.e., accuracy of 63.33). Evidently, in sequence length of 10, 15, and 19, the CNN model could consistently outperform the state-of-the-art baseline by around 4% (see Table 3), highlighting the added value of this model.

Analysing the models revealed that almost all models appeared to have their highest performance on sequence length of 15, and occasionally performed higher or lower on larger or smaller sequence length. One reason for such behaviour could be that in a sequence length of 15, there are enough observations on different game tasks in a game episode to predict the performance of the next game task in the same episode, while smaller sequence lengths might not be fully capable of covering all game tasks due to a lack of data points on different tasks, or larger sequence length may entail insufficient data points on all game tasks in one episode since some players may lose/win a game episode by completing only a few game tasks. Similar to the previous experiment on original dataset of the game, compared to other models, baseline model could not handle sequential data well. And the CNN model seems to be among the best models on the padded dataset since in each game episode, there are certain distinct local patterns, and this model manages to learn those local patterns effectively by assuming limited time/sequence relation depending on previous k instances.

The prediction results from the padded dataset provide an interesting demonstration of the capacities of our proposed deep knowledge tracing approach. Both CNN and LSTM models did well at predicting student responses on different game tasks. Upon comparing the results on the padded dataset with those of original dataset, around 14% gain is observed in prediction accuracies and AUCs. This clearly highlights the importance of considering padding techniques while dealing with sequential tasks. More explicitly, using the padded dataset, models can use data points from players on a specific game episode to predict their performance in the next game task within a game episode.

5.3. Performance of models on the RMA dataset

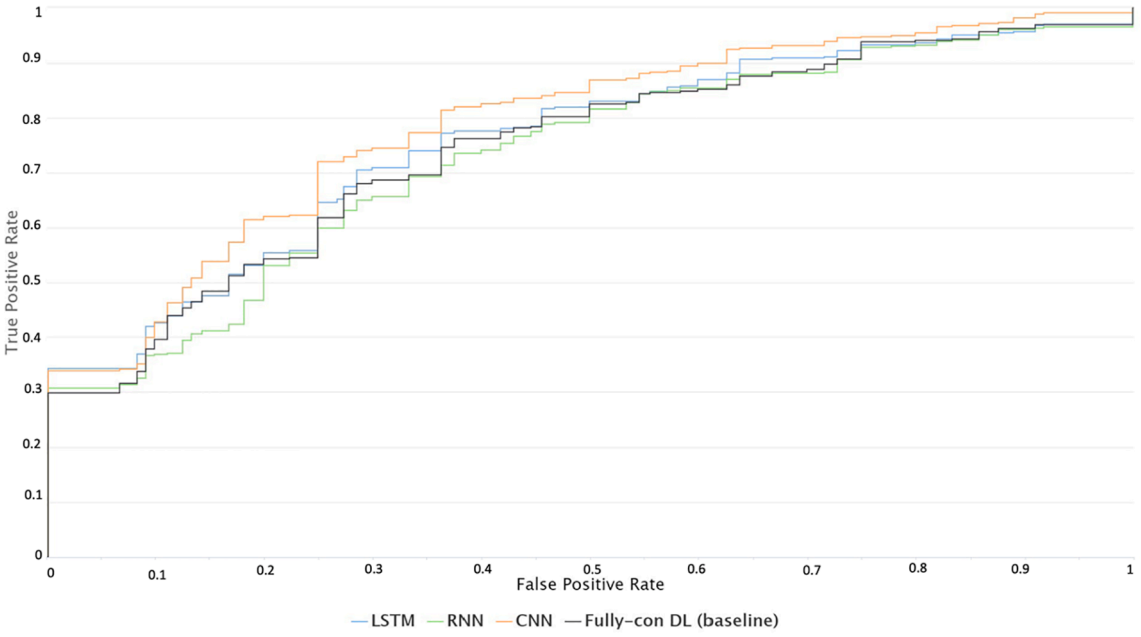
Table 4 illustrates AUCs and squared correlation measures of the cross-validated model predictions on the RMA dataset for different sequence lengths, while Fig. 8 depicts the ROC measures of the models on RMA dataset. Furthermore, Fig. 9 presents accuracy, precision, and recall measures of these models for different sequence lengths. Similar to the original and padded datasets, it is apparent that the sequence length of 15 archives the highest results, while smaller or higher sequence lengths produce rather lower results. The highest and lowest performances were achieved by the CNN and baseline models on sequence lengths 15 and 19, respectively.

More specifically, on the sequence length of 10, the CNN model outperformed other models, with a prediction accuracy of 76.52%, and an AUC of 0.842. With regard to sequence length of 15, CNN followed by RNN and LSTM models resulted in the highest performance with an accuracy of 84.58 and 82.92, respectively. These models also obtained the highest AUC and R^2 scores. On the sequence length of 19, CNN and LSTM models obtained the highest accuracy with 83.33 for CNN and 77.5 for LSTM, while the baseline model appears to have the lowest performance (i.e., accuracy of 66.11). Evidently, in sequence length of 10, 15, and 19, CNN model could consistently outperform the state-of-the-art baseline by up to 10% (see Table 4), highlighting the added value of this model.

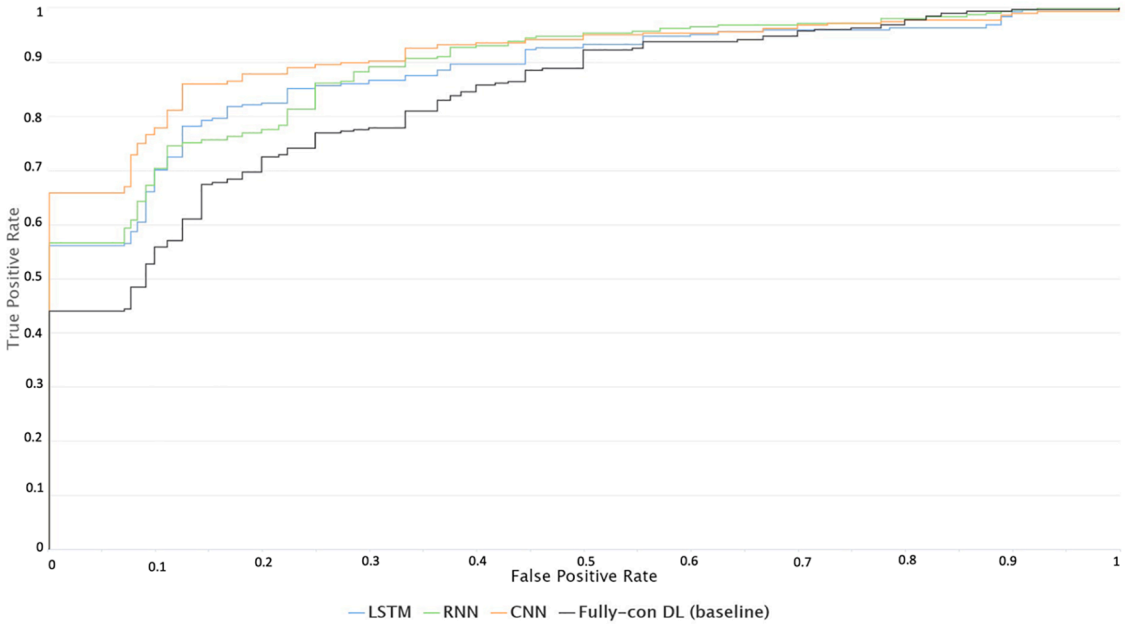
Analysing these models on the RMA dataset revealed that while increment and decrement in sequence length may result in a lower and higher performance in RNN and LSTM models, the CNN model keeps

Table 3
AUC and squared correlation measures for classifiers over three sequence lengths for the PA dataset.

Sequence Length	LSTM		RNN		CNN		MLP (baseline)	
	AUC	R^2	*AUC*	R^2	*AUC*	R^2	*AUC*	R^2
10	0.757	0.167	0.731	0.163	**0.787**	0.180	0.743	**0.203**
15	0.881	0.398	0.890	0.399	**0.913**	**0.444**	0.835	0.233
19	0.680	**0.152**	0.709	0.099	**0.730**	0.109	0.690	0.119



a)

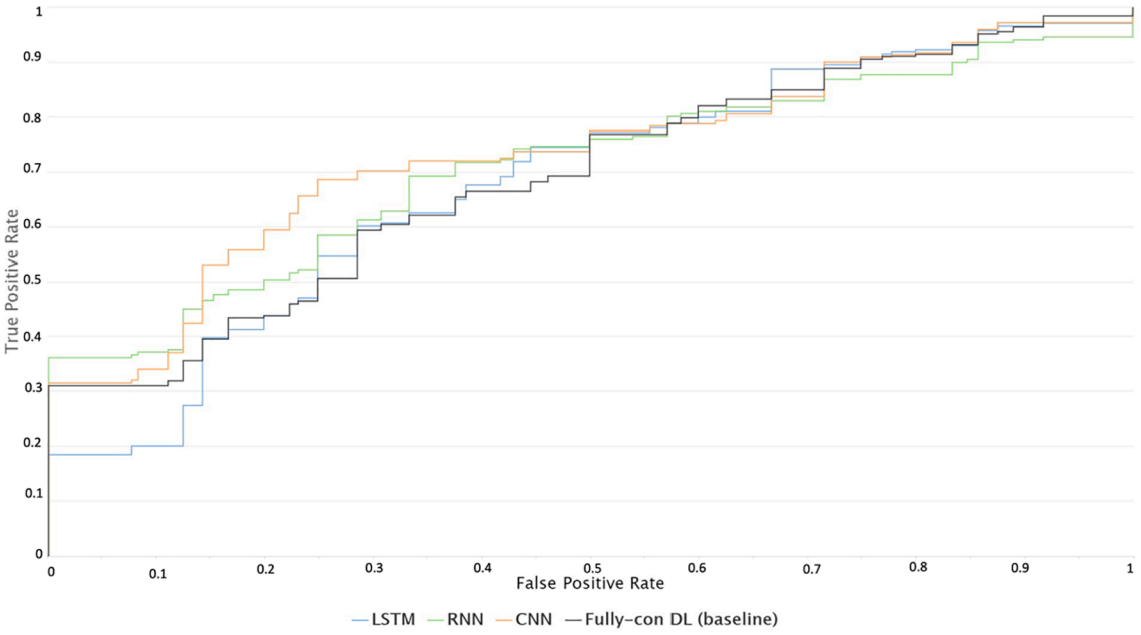


b)

Fig. 6. ROCs of classifiers over: a) 10, b) 15, and c) 19 sequence lengths of actions in PA dataset.

behaving consistently and even shows a better result in larger sequence length. RNN and LSTM models attempt to capture the sequential dependencies from beginning to the end of the sequence (they model sequential data holistically), meaning it assumes that every point in the sequence depends on every previous sequence step. This does not seem to be suitable to our case of game data as more than a few consecutive solution sequences are unnecessary. Thus, CNN model performs better and more stable as it assumes limited time/sequence relation and the same local patterns are related everywhere.

The prediction results from RMA dataset provide an interesting demonstration of the capacities of our proposed deep knowledge tracing approach. The CNN model did well at predicting the student responses on different game tasks. Much like the experiment on padded dataset, upon comparing the results on RMA dataset with those of original dataset, around 14% gain was observed in both prediction accuracies and AUCs. This clearly highlights the importance of considering replacing missing series or padding techniques while dealing with sequential tasks.



c)

Fig. 6. (*continued*).

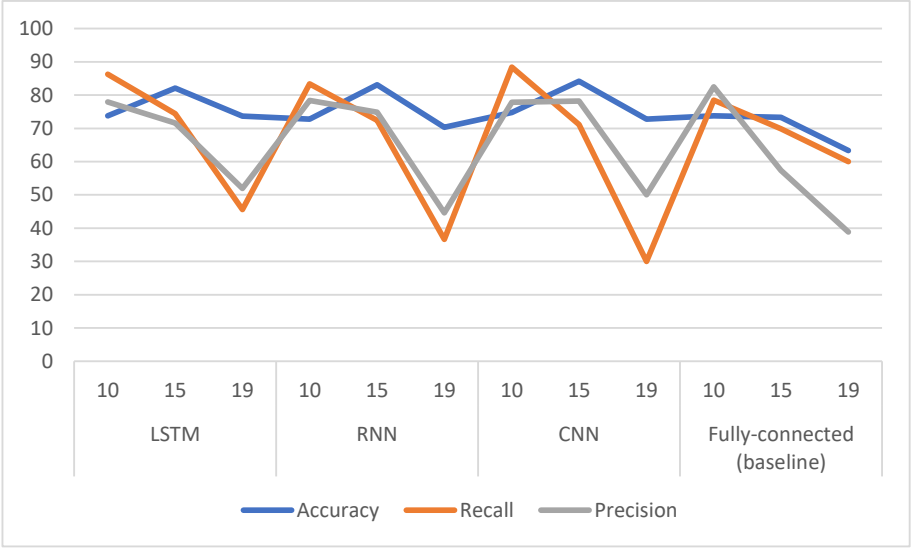


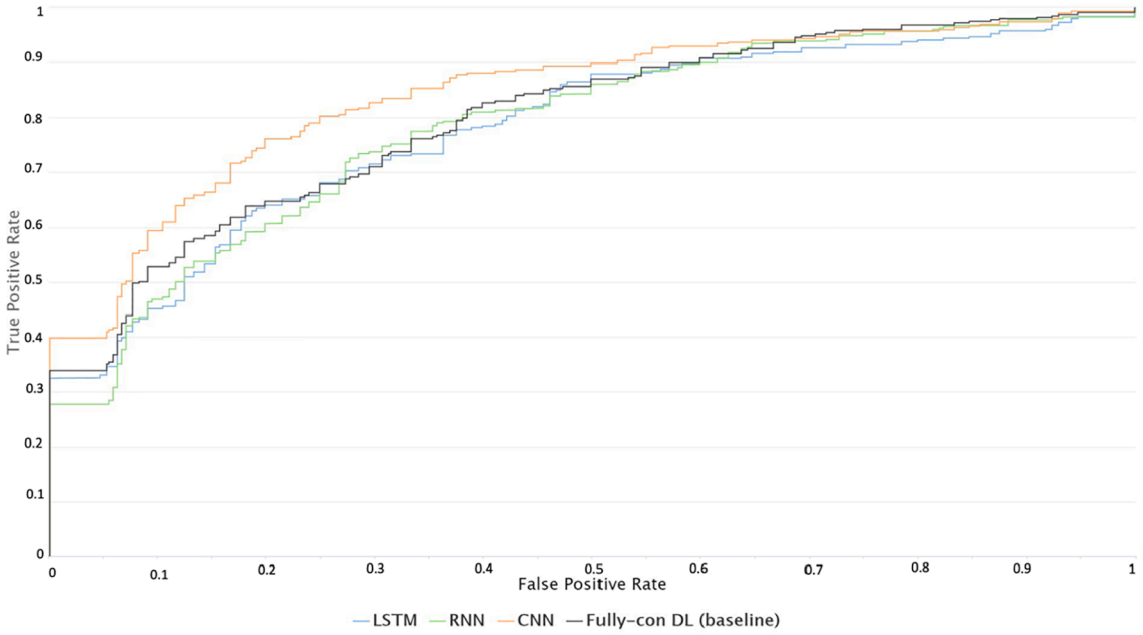
Fig. 7. Accuracy, precision, and recall measures of the models for different sequence lengths on the PA dataset.

Table 4
AUCs and squared correlation measures for classifiers over three sequence lengths for RMA dataset.

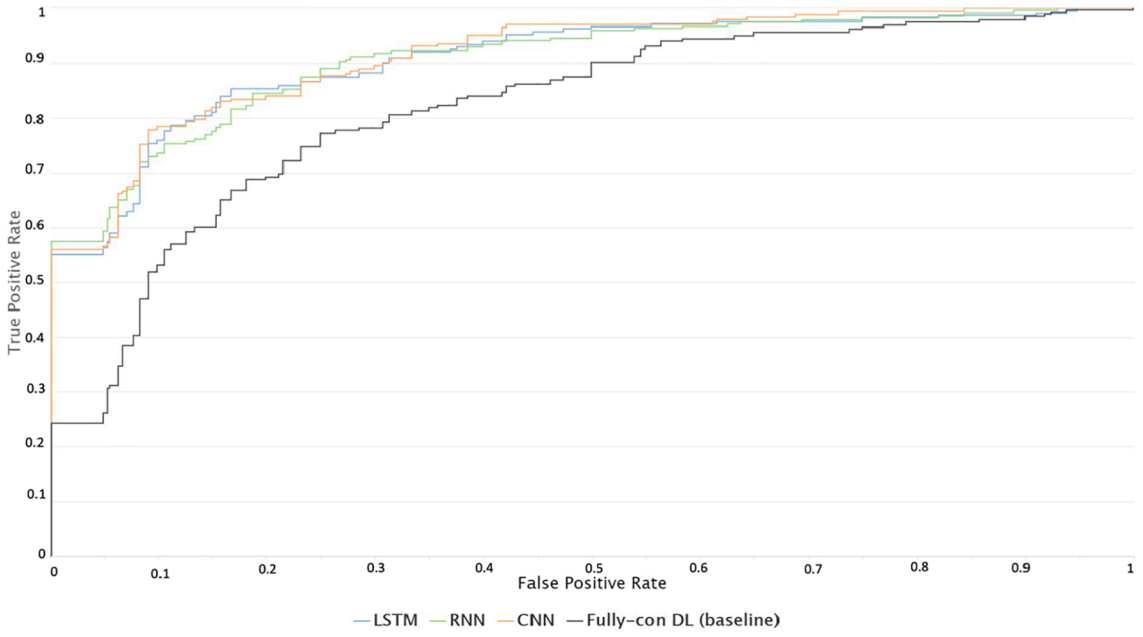
Sequence Length	LSTM		RNN		CNN		MLP (baseline)	
	AUC	R^2	*AUC*	R^2	*AUC*	R^2	*AUC*	R^2
10	0.780	0.180	0.784	0.227	**0.842**	**0.317**	0.799	0.223
15	0.906	0.440	0.906	0.465	**0.913**	**0.490**	0.818	0.219
19	0.811	0.347	0.751	0.160	**0.870**	**0.475**	0.780	0.148

5.4. Performance of models on multi steps forecasting

Table 5 illustrates the averaged performance of cross-validated multi-step model predictions (i.e., predicting 2, 3, 4, and 5 steps ahead) through AUCs and squared correlation measures on the RMA dataset for a sequence length of 15. Furthermore, Fig. 10 presents accuracy, precision, and recall measures of these models. We opted to employ the CNN model and a sequence length of 15 as, in our previous



a)

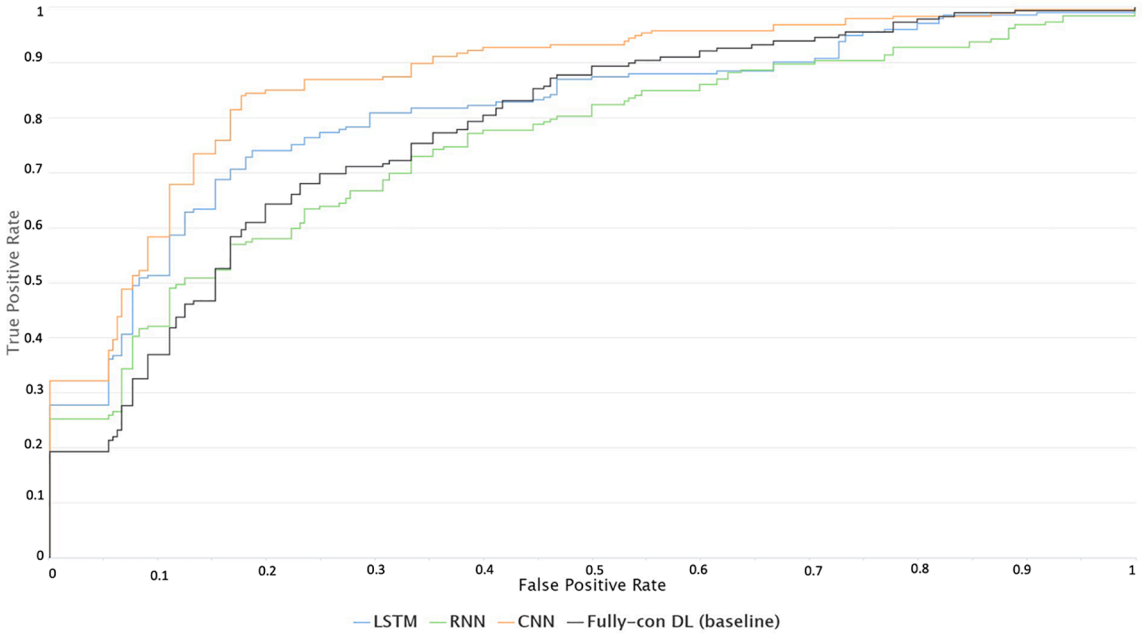


b)

Fig. 8. ROCs of classifiers over: a) 10, b) 15, and c) 19 sequence lengths of actions in RMA dataset.

experiments, they revealed to be the highest performing model and the most optimum sequence length; needless to say that the baseline model has also been included.

According to the results, increment in the prediction steps results in decrease of model performances. Overall, the CNN model outperformed the baseline model. The highest performance of CNN model on these multi-steps belong to prediction of students' performance two steps ahead, with 74.6% accuracy and 0.831 AUC, whereas the lowest performance was obtained by the baseline model on prediction of five steps ahead (i.e., accuracy of 59.17%). The CNN model could successfully predict performance of students on game tasks for up to four steps ahead with accuracy of more than 70%. On the other hand, the baseline model could not obtain a prediction accuracy better than 64% while predicting four steps ahead. The best performance of the baseline model was achieved while predicting two steps ahead with an accuracy of roughly 68%. This clearly shows that while the CNN model can consistently predict with high accuracy the performance of students in the upcoming game tasks (for up to four steps ahead), the baseline model falls short



c)

Fig. 8. (*continued*).

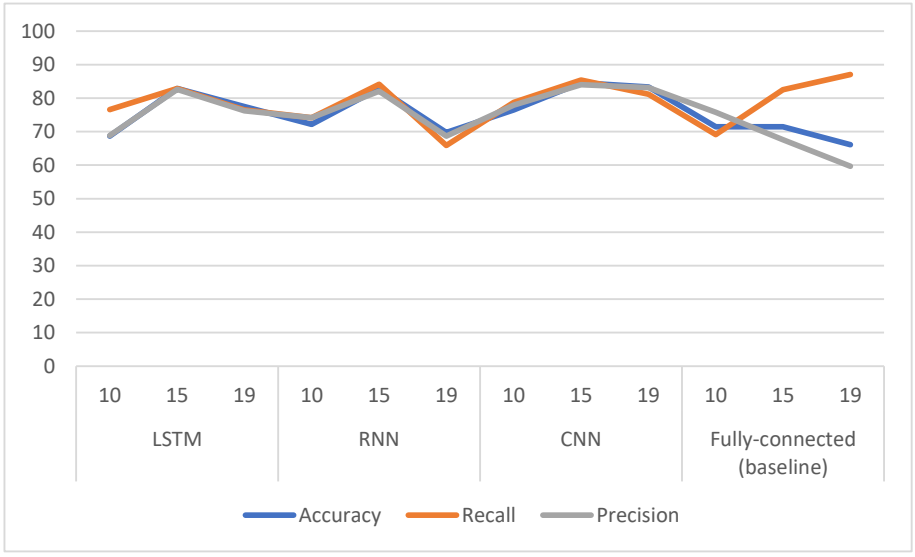


Fig. 9. Accuracy, precision, and recall measures of the models for different sequence lengths on the RMA dataset.

Table 5
AUCs and squared correlation measures of classifiers on multi-step predictions over sequence length of 15.

Dataset	CNN								MLP (baseline)							
	AUC				R^2				*AUC*				R^2			
	2	*3*	*4*	*5*	*2*	*3*	*4*	*5*	*2*	*3*	*4*	*5*	*2*	*3*	*4*	*5*
15 RMA	***0.831***	***0.801***	***0.760***	***0.666***	**0.272**	**0.247**	**0.169**	**0.067**	0.744	0.740	0.702	0.636	0.134	0.113	0.085	0.032

when it comes to prediction of more than one step ahead.

Surprisingly, the prediction accuracy of CNN model is still higher while predicting two steps ahead compared to one step ahead in the baseline model on the same dataset. The results of the multi-step prediction from the RMA dataset provide an interesting demonstration of the capacities of our proposed deep knowledge tracing approach.

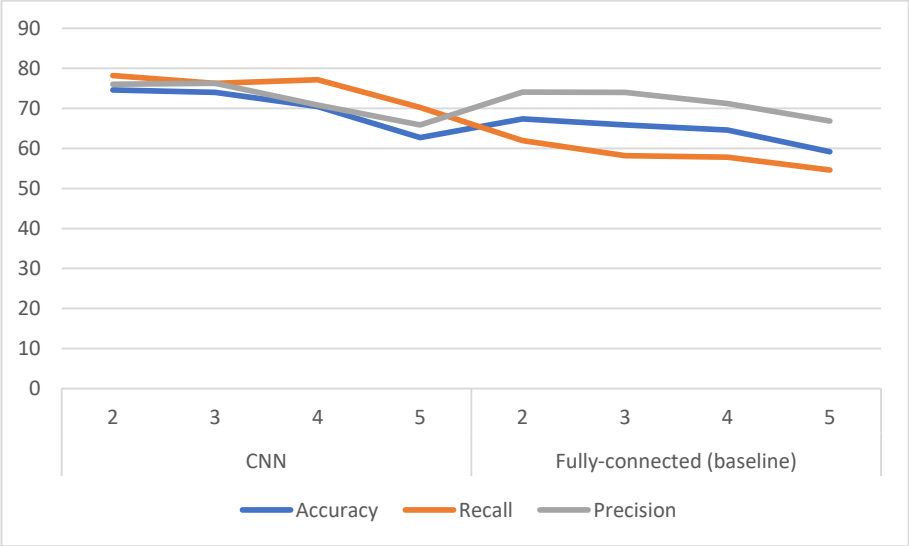


Fig. 10. Accuracy, precision, and recall measures of the CNN and baseline models for multi-step model predictions.

6. Discussion and conclusions

In this research, we proposed an educational data mining approach for modelling or tracing the players' knowledge by predicting their future performance based on their past activity in educational computer games, using emerging deep knowledge tracing algorithms. Our proposed approach, GameDKT, integrates state-of-the-art machine learning methods with domain knowledge to predict the players' performance in the upcoming game tasks.

Results of our experiments reveal that GameDKT approach could successfully predict the players' performance in the upcoming game tasks using the cross-validated CNN model, which has an accuracy and AUC of roughly 85% and 0.913, respectively. Upon comparing the results of this model with those of baseline model, up to 14% gain is observed in favour of CNN model. More specifically, depending on the nature of datasets and sequence of players' actions, CNN model consistently outperforms the baseline model. For most of the models, the results obtained on RMA and PA datasets on sequence length of 15 were much higher than those on the OA dataset (the original dataset) with sequence lengths 10 or 19 as the RMA and PA use stationary sequence of data points from one game and player to predict the performance of the same player on the next task in the same game episode. The reason for 15 being the most optimal sequence length of actions is that the average number solutions developed by all the players is around 15. This signals the importance of considering replacing missing series or padding techniques while dealing with sequential tasks as well as identifying proper sequence lengths of actions.

Generally, the baseline model appears to have the lowest performance compared to other models, and an increment in sequence length of actions result in decrease in its prediction accuracy. Contrary to our expectations and results of previous research (e.g., Nakagawa, Iwasawa, & Matsuo, 2019; Piech et al., 2015; Wang, Sy, Liu, & Piech, 2017b; Xiong et al., 2016), the CNN model appears to have a better performance compared to sequential models such as RNN or LSTM over the three sequence lengths in all three datasets. While occasionally showing a prediction power as good as the CNN model on some datasets in longer sequence lengths of actions (i.e., 15 or 19), compared to the CNN and RNN models, the LSTM sometimes falls short on shorter sequence lengths (10 in particular). On the other hand, the RNN model seems to have a better performance on shorter sequences of actions compared to some of the longer sequence lengths. RNN seems to have outperformed LSTM on some specific sequence lengths since it usually only looks at a few latest inputs (short-term modelling) and has the power to remember exact patterns of sequential data, while LSTM keeps, in its memory, some single observations for a long time, which might not necessarily be useful in this case. One reason for the occasional performance deficit of LSTM model could be that in this type of predictions, many-to-one where there is a class as an output with a sequence of multiple steps as input, the lagged observations are rather close to the time being predicted. Another point to note is that the current task merely deals with fixed sequence lengths (i.e., 10, 15, and 15), and LSTM is mostly known to achieve good results by learning to predict in non-stationary sequences or time series. Essentially, RNN and LSTM models attempt to capture sequential dependencies from the beginning to the end of the sequence (model sequential data holistically) or discover periodic intervals from the dataset, meaning it assumes that every point in the sequence depends on every previous sequence step. However, in the case of our game data, more than a few consecutive solution sequences seem to be unnecessary since predicting the performance of the next sequence step does not necessarily require knowing the whole preceding solutions.

Although not specifically developed for non-image data, CNNs could achieve state-of-the-art results on problems with time series or sequential data analysis. This model outperformed other models since it seems to be better at identifying local patterns (considering that in each game, there is a certain distinct local pattern and the aim is to learn those local patterns effectively) due to its main assumption that the same local patterns are related everywhere, and the fact that it does not develop a relationship between hidden vectors of each sequence step. Also, it is faster and computationally lighter compared to the other models (LSTM or RNN) as there are fewer sequential calculations. Even though it might not be fully correct to compare our findings with previous research due to several reasons, including type of the tasks and sequence lengths, our results are in line with those reported by Nabi, Tahmid, Rafi, Kader, and Haider (2021), where the CNN model achieved a higher performance compared to recurrent neural networks when forecasting on small number of features and data amount.

When it comes to multi-step predictions, i.e., predicting performance of players in 2, 3, 4, and 5 steps ahead, on sequence lengths of 15, our findings reveal that the CNN model can predict performance of students on game tasks for up to four steps ahead with an accuracy of more than 70% (76% for two steps ahead). This clearly indicates the added value of this model over the baseline model that could not obtain a prediction accuracy better than 64% while predicting four steps ahead. This result signals that while the CNN model can consistently predict with high accuracy the performance of students in the upcoming game tasks (for

up to four steps ahead), the baseline model falls short when it comes to predictions of more than one step ahead.

From an educational point of view, development of such an approach could be especially useful for all types of educational computer games as no formal or informal testing would be necessary when the learners' skill or knowledge undergoes constant assessment. Such continuous assessment could potentially provide learners with the most optimal sequence of learning tasks or items. For example, after finishing 10, 15, or 19 game tasks, our proposed approach can predict the learners' performance in game tasks 11, 16, and 20, respectively. Moreover, since our proposed approach can reach a high accuracy in predicting the performance of learners for up to four upcoming game tasks, in the case of sequence of 10 and 15 actions, we can calculate their expected knowledge state on game tasks 11 to 14 and 16 to 19, respectively. This would allow us to test the possible upcoming game tasks to learners and compute their expected knowledge state given that choice. Furthermore, such knowledge tracing would allow to produce human-like action sequences by predicting the succeeding learners' actions and performances, which can be used to design adaptive NPCs to enhance game engagement and avoid player churn. These findings are consistent with those reported by Kantharaju et al. (2018) and Cui et al. (2019). However, GameDKT performs better than the other approaches due to several reasons including the fact that it benefits from deep learning-based approaches that effectively deal with sequential data in modelling latent information of learners.

A few limitations of this study could be regarded as directions for future research. For example, even though we have effectively modelled the learners' knowledge and accurately traced their skills, the dataset used in this research was limited to one game. Therefore, it would be useful if in the future, our proposed approach can be applied on datasets from different educational computer games to assess its generalizability. Another limitation of this research is that our proposed approach has not yet been put into practice and its effectiveness in practice is unknown. Another future work could be to integrate the approach into AutoThinking game to provide adaptive game task sequence generation, NPCs, feedback, etc. Finally, including side information on learners in the process of knowledge tracing could potentially improve its performance. Thus, we also plan to take advantage of more predictive attributes associated with learners and the game to boost the performance of the proposed approach.

CRediT authorship contribution statement

Danial Hooshyar: Conceptualization, Methodology, Supervision, Visualization, Writing – review & editing. **Yueh-Min Huang:** Conceptualization, Supervision, Writing – review & editing. **Yeongwook Yang:** Conceptualization, Data curation, Formal analysis, Methodology, Supervision, Visualization, Writing – review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. 2021R1C1C2004868).

References

Abyaa, A., Idrissi, M. K., & Bennani, S. (2019). Learner modelling: Systematic review of the literature from the last 5 years. *Educational Technology Research and Development, 67*, 1105–1143.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1–6). IEEE.

Corbett, A. T., & Anderson, J. R. (1994). Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-adapted Interaction, 4*, 253–278.

Cui, Y., Chu, M.-W., & Chen, F. (2019). Analyzing student process data in game-based assessments with Bayesian knowledge tracing and dynamic Bayesian networks. *Journal of Educational Data Mining, 11*, 80–100.

El Mawas, N., Hooshyar, D., & Yang, Y. (2020). Investigating the learning impact of autothinking educational game on adults: A case study of France. *CSEDU, 2*, 188–196.

Falcão, T. P., e Peres, F. M.d. A., de Morais, D. C. S., & da Silva Oliveira, G. (2018). Participatory methodologies to promote student engagement in the development of educational digital games. *Computers & Education, 116*, 161–175.

Gweon, G.-H., Lee, H.-S., Dorsey, C., Tinker, R., Finzer, W., & Damelin, D. (2015). Tracking student progress in a game-like learning environment with a Monte Carlo Bayesian knowledge tracing model. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge* (pp. 166–170).

Heathcote, A., Brown, S., & Mewhort, D. J. K. (2000). The power law repealed: The case for an exponential law of practice. *Psychonomic Bulletin & Review, 7*, 185–207.

Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. In *Neural networks for perception* (pp. 65–93). Elsevier.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*, 504–507.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780.

Hooshyar, D., Lim, H., Pedaste, M., Yang, K., Fathi, M., & Yang, Y. (2019). AutoThinking: An adaptive computational thinking game. In *International conference on innovative technologies and learning* (pp. 381–391). Cham: Springer.

Hooshyar, D., Malva, L., Yang, Y., Pedaste, M., Wang, M., & Lim, H. (2021). An adaptive educational computer game: Effects on students' knowledge and learning attitude in computational thinking. *Computers in Human Behavior, 114*, Article 106575.

Hooshyar, D., Pedaste, M., Saks, K., Leijen, Ä., Bardone, E., Wang, M. J. C., & Education. (2020). Open learner models in supporting self-regulated learning in higher education: A systematic literature review. *Computers & Education, 154*, 103878.

Hooshyar, D., Yousefi, M., & Lim, H. (2018). Data-driven approaches to game player modeling: A systematic literature review. *ACM Computing Surveys, 50*, 1–19.

Huo, Y., Wong, D. F., Ni, L. M., Chao, L. S., & Zhang, J. (2020). Knowledge modeling via contextualized representations for LSTM-based personalized exercise recommendation. *Information Sciences, 523*, 266–278.

Kantharaju, P., Alderfer, K., Zhu, J., Char, B., Smith, B., & Ontanón, S. (2018). Tracing player knowledge in a parallel programming educational game. *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Khajah, M., Lindsey, R. V., & Mozer, M. C. (2016). How deep is knowledge tracing?.

Khajah, M., Wing, R., Lindsey, R., & Mozer, M. (2014). Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. *Educational Data Mining 2014: Citeseer*.

Kim, S., Kim, W., Jung, H., & Kim, H. (2021). DiKT: Dichotomous Knowledge Tracing. In *International Conference on Intelligent Tutoring Systems* (pp. 41–51). Springer.

Liu, S., Zou, R., Sun, J., Zhang, K., Jiang, L., Zhou, D., & Yang, J. (2021). A hierarchical memory network for knowledge tracing. *Expert Systems with Applications, 177*, Article 114935.

Nabi, K. N., Tahmid, M. T., Rafi, A., Kader, M. E., & Haider, M. A. (2021). Forecasting COVID-19 cases: A comparative analysis between Recurrent and Convolutional Neural Networks. *Results in Physics, 24*, Article 104137.

Nakagawa, H., Iwasawa, Y., & Matsuo, Y. (2019). Graph-based knowledge tracing: modeling student proficiency using graph neural network. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)* (pp. 156–163). IEEE.

Osika, A., Nilsson, S., Sydorchuk, A., Sahin, F., & Huss, A. (2018). Second language acquisition modeling: An ensemble approach. *arXiv preprint arXiv:1806.04525*.

Pelánek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: An overview of learner modeling techniques. *User Modeling and User-Adapted Interaction, 27*, 313–350.

Piech, C., Spencer, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L., & Sohl-Dickstein, J. (2015). Deep Knowledge Tracing.

Schodde, T., Bergmann, K., & Kopp, S. (2017). Adaptive robot language tutoring based on Bayesian knowledge tracing and predictive decision-making. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 128–136).

Su, Y., Liu, Q., Liu, Q., Huang, Z., Yin, Y., Chen, E., … Hu, G. (2018). Exercise-enhanced sequential modeling for student performance prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*.

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems, 39*, 43–62.

Wang, Z., Feng, X., Tang, J., Huang, G. Y., & Liu, Z. (2019). Deep knowledge tracing with side information. In *International conference on artificial intelligence in education* (pp. 303–308). Springer.

Wang, L., Sy, A., Liu, L., & Piech, C. (2017a). Deep knowledge tracing on programming exercises. In *Proceedings of the fourth (2017) ACM conference on learning@ scale* (pp. 201–204).

Wang, L., Sy, A., Liu, L., & Piech, C. (2017b). *Learning to Represent Student Knowledge on Programming Exercises Using Deep Learning*. International Educational Data Mining Society.

Whitney, K. L. (2021). *The effect of personalized learning on student achievement*. Regent University.

Xiong, X., Zhao, S., Van Inwegen, E. G., & Beck, J. E. (2016). *Going deeper with deep knowledge tracing*. International Educational Data Mining Society.

Yeung, C.-K. (2019). Deep-IRT: Make Deep Learning Based Knowledge Tracing Explainable Using Item Response Theory.

Yudelson, M. V., Koedinger, K. R., & Gordon, G. J. (2013). Individualized Bayesian knowledge tracing models. In *International conference on artificial intelligence in education* (pp. 171–180). Springer.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhang, J., Shi, X., King, I., & Yeung, D.-Y. (2017). Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th international conference on World Wide Web* (pp. 765–774).