KT-Bi-GRU: Student Performance Prediction with a Recurrent Knowledge Tracing Neural Network

Marina Delianidi* and Konstantinos Diamantaras†, *Senior member, IEEE,*
Department of Information and Electronic Engineering
International Hellenic University, Sindos, 57400, Greece
Email: *d.marina@iee.ihu.gr, †kdiamant@ihu.gr

Abstract

Student performance is affected by their knowledge which changes dynamically over time. Therefore, employing recurrent neural networks (RNN), which are known to be very good in dynamic time series prediction, can be a suitable approach for student performance prediction. We propose such a neural network architecture containing two modules: (i) a dynamic sub-network including a recurrent Bi-GRU layer used for knowledge state estimation (ii) a non-dynamic, feed-forward sub-network for predicting answer correctness based on the current question and current student knowledge state. The model modifies our previously proposed architecture and is different from all other existing models, because it estimates the student's knowledge state considering only their previous responses and not the current question to be answered. The advantage of the new approach is that the knowledge state vector generated by the dynamic sub-network can be matched against any other question and thus the model could be used for other purposes as well, such as educational content recommendation. We studied both single-skill and multi-skill question scenarios and employed embeddings to represent questions and responses. In the multi-skill case the initialization of the question embedding matrix with pretrained word-embeddings is found to improve model performance. The experimental results showed that our current KT-Bi-GRU model and the previous one have similar performance while both surpassed the performance of previous state-of-the-art knowledge tracing models for six out of seven datasets where in some cases, the difference is quite noticeable.

Index Terms

Knowledge tracing, student performance prediction, dynamic neural networks, knowledge state.

KT-Bi-GRU: Student Performance Prediction with a Recurrent Knowledge Tracing Neural Network

I. Introduction

Knowledge tends to change with time and this dynamic property has been a central subject of study in the educational data mining field in recent decades. Practically, the student's knowledge changes in a positive way when the student learns a learning object (i.e. a concept or skill) or in a negative way when the student forgets. Maintaining, over time, the probability that the student has mastered various learning objects is called Knowledge Tracing (KT) [1]. The learning objects in a specific area, for example, "Algebra", "Geometry", "Physics" etc, are called knowledge components (KC) and compose the picture of the knowledge state (KS) of the student. The process of estimating students' performance helps to assess their knowledge state and can contribute to the future improvement of their learning performance by tracing the current knowledge. During the learning process through an electronic Intelligent Tutoring System (ITS), students' activities are implicitly recorded in logs as they interact with the system. These log files can be used during the knowledge tracing process in order to assess the evolution of the student knowledge state over time.

In general, according to [2], knowledge tracing approaches can be classified into three main categories: (i) Models based on statistical or probabilistic methods such as the BKT [1] which is implemented through the Hidden Markov model; (ii) Logistics models such as LFA [3] and PFA [4] and (iii) Deep Learning models such as DKT, DKMVN, Deep-IRT [5], [6], [7]. Recently, methods from the first two categories were combined with deep learning methods, for example DBN [8] and DPFA [9], achieving improved results. Additionally, convolutional neural networks [10], [11], [12], [13] and graph neural networks models [14], [15] have been recently proposed for the KT task. The aim of all the works is the optimal representation of the knowledge state and the prediction of student performance so that the learning process can be improved and adapted to the student's learning needs.

In the basic setup, a student interacts with an ITS and gives answers to questions q_i. The variable $r_i \in \{0, 1\}$ indicates the correctness of the answer, where 1 means correct and 0 means wrong answer. At each time instance, the student's KS is formed in relation to the skill that is examined based on the correct or incorrect answer. The questions are related to one or more skills taken from some skill set $S = \{s_1, s_2, \ldots, s_n\}$. Our prediction task is formalized as follows: given the student's past interaction sequence $X = (x_1, x_2, \ldots, x_i, \ldots, x_{t-1})$, where $x_i = \{q_i, r_i\}$ is the interaction pair at the i-th time instance, we must accurately predict the correctness r_t of the answer to the current question q_t. An example of the prediction task based on students' interactions

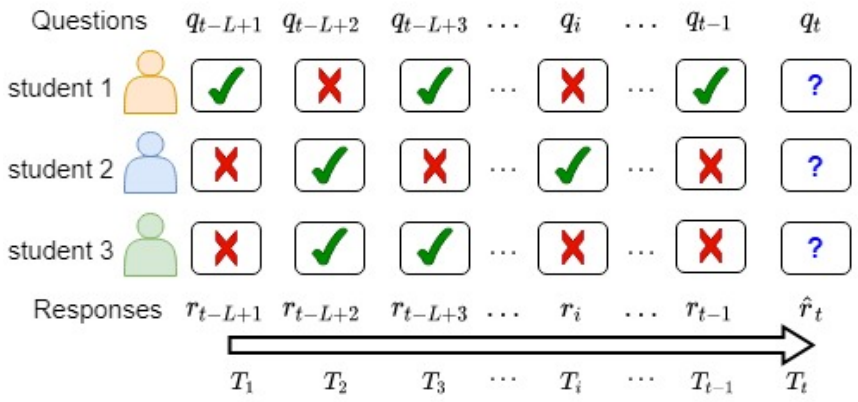


Fig. 1. Prediction task: based on the history window with length L of the previous interactions, the response correctness r_t to the question q_t at the current time instance t must be predicted for every student.

can be seen in Figure 1. A more accurate prediction $\hat{r_t}$ corresponds to a better assessment of the student's knowledge level in different skills at the time when the question q_t will be asked.

The student knowledge state at time t can be determined by the correctness r_i of the responses to the previous questions $q_i, i = 1, \ldots, t-1$, and should not be a function of q_t since we typically do not know the response to this question. However, in prior works [1], [4], [5], [6], [7], [16] employing neural networks for student performance prediction, the current question q_t is not separated from the previous ones. Even though, using this approach, it is still possible to predict r_t, it is difficult to identify a part of the model that represents the current student knowledge state. In our earlier work [16] we followed this mainstream approach proposing a deep neural model based on a Bi-directional Gated Recurrent Unit layer.

In this work we propose to model r_t taking into account the student's KS up to the time instance $t-1$. To this end, we take two steps: first, we separately estimate a representation vector $\mathbf{v}_t$ of the current KS using a recurrent neural layer using the student interaction history up to time $t-1$ and without knowing q_t. Then we combine $\mathbf{v}_t$ with the representation of q_t and apply a feed-forward classification neural network to actually estimate r_t. The advantage of this approach is that having $\mathbf{v}_t$ we can combine with any question q and predict the correctness of the student's answer to q at any time t. This allows us to monitor the student's performance in questions relating to specific skills and identify student's weaknesses. Potentially this model could be used for other important tasks such as building an educational recommendation system. The proposed model is tested on seven public datasets with single-skill or multi-skills questions in the specific area of mathematics.

The contributions of this work can be summarized as follows:

1) We propose a new neural network architecture KT-Bi-GRU incorporating a bi-directional recurrent layer whose output represents the student knowledge state based on his/her previous interactions. The recurrent layer output is used in combination with the current question to predict the correctness of the student response.
2) We studied the initialization of question embed-

dings by comparing pretrained vs. random vectors. We found that in the case of multi-skill questions, the initialization using pretrained Word2Vec embeddings taken from the verbal description of the skills involved in the questions contributes to a better model performance as opposed to random initialization. This is not observed in the case of single-skill questions.

3) We compare the proposed model performance against earlier, state-of-the art neural networks, including our own earlier Bi-GRU RNN model. The experimental evaluation is performed on seven different datasets.
4) The proposed architecture facilitates the estimation of the student knowledge state, a feature which could be potentially useful for tasks such as student clustering or educational content recommendation.

The rest of this paper is organized as follows. Section II provides the literature review of the existing student performance prediction models using the KT task specifically based on deep learning techniques. In Section III, referring to our previous work [16], we present the new deep learning based knowledge tracing model with differentiated inputs. The datasets we used in our experiments are described in Section IV, while the experimental settings and parameters are presented in Section V. The experimental results, in comparison to the deep learning based state-of-art models DTK, DKVMN and Deep-IRT, are discussed in Section VI. We conclude the paper and present the future work in Section VII.

II. Literature Review

The tracing of student knowledge using Bayesian Networks has been introduced by [1] and it is referred to as Bayesian Knowledge Tracing (BKT). The method belongs to the probabilistic knowledge modeling techniques. Due to the continuous development and wide use of e-learning there has been an increasing interest in this topic which resulted in the development of a variety of approaches. The so called logistic KT methods including Learning Factor Analysis (LFA) [3] and Performance Factor Analysis (PFA) [4] have been shown to achieve better performance compared to the BKT model. In logistic KT models the probability of a correct answer is represented by a logistic function involving student and knowledge components (KC) parameters.

Deep Neural Networks (DNN) and, especially, Recurrent Neural Networks (RNN) have contributed to the development of the most efficient knowledge tracing models to date. The first knowledge tracing model utilizing RNNs, and specifically the LSTM model, was DKT [5] introduced in 2015. Having as input the one-hot encoded skill tags and the associated responses, the neural network is trained to predict the correctness of the next student's response. The hidden state of LSTM can be considered as the latent state of a student knowledge and can transfer the information of previous interactions to the output level. The output level of DKT, (depending on the question or the skill), estimates the probability of answering the question related to a specific knowledge component correctly.

The Dynamic Key Value Memory Network (DKVMN) [6], is another approach that uses a modified memory augmented neural network (MANN) [17] for knowledge tracing, attempting to capture the relationship between different concepts. The DKVMN model outperforms the DKT model. To encode students' knowledge state, DKVMN uses memory slots as key-value pairs in which learning or forgetting a particular skill are controlled through read and write operations. The concepts are stored in the key component which is fixed during testing, while the value component is updated when a concept state changes. This means that when a student masters a concept in a test, the value component is up-

dated based on the correlation between the corresponding concept and the exercises.

Recently the DKVMN model has been extended by the Deep-IRT model [7]. In Deep-IRT the capabilities of the DKVMN are combined with the Item Response Theory in order to measure both student ability and question difficulty. Another model, named Sequential Key-Value Memory Networks (SKVMN) [18], combines DKVMN with Hop-LSTM, a variation of the LSTM architecture. SKVMN utilizes the Hop-LSTM ability to discover sequential dependencies between exercises but it skips some LSTM cells to approach previous concepts that are considered relevant. In this way, SKVMN tried to overcome the problem of DKVMN to capture long term dependencies on the sequences of exercises and generally on sequential data. The attention mechanism [19] has been also used in other KT models. One of such model is the Self Attentive Knowledge Tracing (SAKT) [20]. SAKT consists of three layers: an embedding layer for interactions and questions followed by a self-attention mechanism layer and a feed-forward layer for student response prediction.

Other KT approaches combine earlier proposed models with deep learning techniques to enhance them with the ability to dynamic knowledge modeling. For example, the Dynamic BKT (DBKT) [8], introduced in 2017, models knowledge by taking into account the correlations between KCs and predicting students' performance based on performance in previous relevant KCs. Similarly, the Deep Performance Factors Analysis (DPFA) [9] model published in 2021, combines the PFA with deep learning models in order to improve it and is summarized as a logistic regression model based on the affinity of previous and future items.

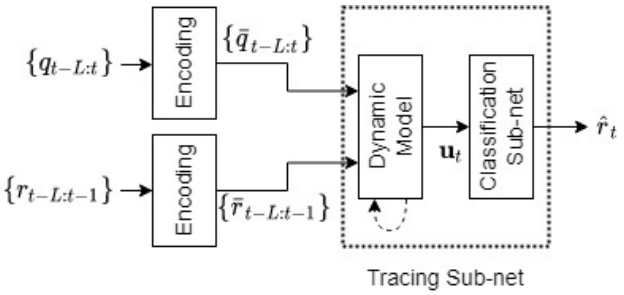
Recent research has presented knowledge tracing models using convolutional neural networks (CNN) [10], [11], [12], [13] and graph neural networks (GNN) [14], [15]. Other works [21], [22] employ CNN and GNN models incorporating recurrent neural layers, such as LSTM, in order to achieve better performance.

In our previous work [16] we proposed a dynamic KT model is using a special type of Recurrent Neural Network called Bidirectional Gated Recurrent Unit (Bi-GRU). Comparing the GRU network [23] with the LSTM network, the GRU has fewer parameters although it typically has similar performance. Bidirectional RNNs [24], such as the Bi-GRU, connect two hidden layers of opposite directions at the same output. This structure provides information to the output layer from the future states (backward direction) and from the past state (forward directions) at the same time. The output of the Bi-GRU network combines and normalizes the outputs of the forward and backward hidden layers at each instant. In this paper we give a short description of our previous work and present a new dynamic KT model named KT-Bi-GRU. The Bi-GRU neural network is a basic component of our both models architecture.

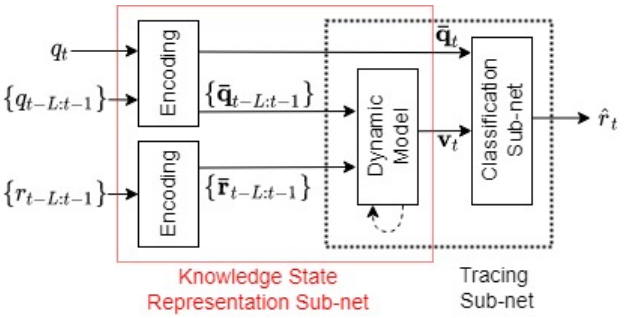
III. Dynamic neural network architecture

As mentioned earlier, knowledge is formed dynamically over time. Recurrent neural networks have been known to be very good at modeling and predicting dynamic processes and so they can be used to model the students' knowledge state. Student response prediction should take into account information regarding the estimated student knowledge state. Since responses can be either correct (1) or wrong (0) we approach the task of predicting them as a binary classification problem. Our model architecture consists of the following parts:

- the input sequences $\{q_{t-L}, \ldots, q_t\}$ and $\{r_{t-L}, \ldots, r_{t-1}\}$ where q_i denotes the question id at time i and r_i denotes the correctness of the response at i. The hyper-parameter L indicates



(a) The early Bi-GRU model architecture [16]



(b) The KT-Bi-GRU model architecture.

Fig. 2. The Bi-GRU models' general architectures. The vector $\mathbf{v}_t$ represents the student's Knowledge State at the current time instance.

the length of the time window used to predict the response r_t.

- an *Encoding* component incorporating two embedding layers which produce vector representations $\bar{\mathbf{q}}_i$, $\bar{\mathbf{r}}_i$ for q_i and r_i, respectively.
- a *Dynamic* component which traces the student's knowledge state using the time sequences $\{\bar{\mathbf{q}}\}$ and $\{\bar{\mathbf{r}}\}$
- a *Classification* component which predicts the student response based on the output of the dynamic component.

The above architecture does not specify an important detail: whether the current question, q_t, should participate in the sequence that feeds the dynamic tracing component. In our earlier work [16] we followed the above design philosophy with q_t actually feeding the dynamic component (Fig. 2a). However, in this case it is difficult to associate the output of the dynamic component with the student's knowledge state at time t since the current KS should not depend on the current question q_t which is not answered yet.

For this reason we propose here an alternative architecture depicted in Figure 2b where the current question does not feed the dynamic tracing component but is only used as input for the classification component in order to predict r_t. The architectures of the two models are described in detail below.

A. Early Dynamic Bi-GRU Model

The architecture of our earlier proposed recurrent model Bi-GRU[1] [16] is shown in Figure 2a. The correctness r_t of the answer at time t is defined as a function of

[1]https://github.com/delmarin35/Dynamic-Neural-Models-for-Knowledge-Tracing

the student's previous interactions $(q_i, r_i), i = t-1, t-2, \ldots,$ and the current question q_t:

$$r_t = \phi(q_t, q_{t-1}, q_{t-2}, \ldots, r_{t-1}, r_{t-2}, \ldots) + \epsilon_t \quad (1)$$

where ϵ_t is the prediction error.

In order for the model to remember arbitrarily long sequences the Dynamic component of the architecture incorporates a recurrent Bidirectional GRU (Bi-GRU) layer. A Bi-GRU layer consists of two Gated Recurrent Unit (GRU) layers, one for the forward time direction and another for the backward time direction (Fig. 3).

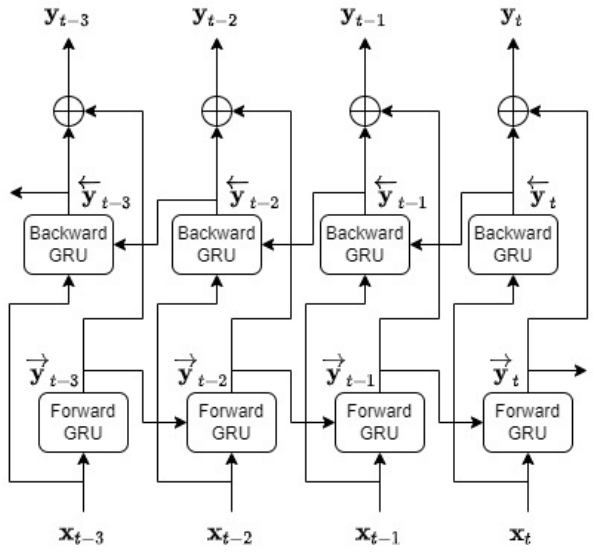


Fig. 3. A Bidirectional GRU layer.

The forward GRU layer is described by the following equations:

$$\overrightarrow{\mathbf{y}}_t = (1 - \mathbf{z}_t) \circ \overrightarrow{\mathbf{y}}_{t-1} + \mathbf{z}_t \circ \mathbf{h}_t \quad (2)$$

$$\mathbf{h}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h [\mathbf{r}_t \circ \overrightarrow{\mathbf{y}}_{t-1}] + \mathbf{b}_h) \quad (3)$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \overrightarrow{\mathbf{y}}_{t-1} + \mathbf{b}_z) \quad (4)$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \overrightarrow{\mathbf{y}}_{t-1} + \mathbf{b}_r) \quad (5)$$

where $\mathbf{z}$ is the update gate vector, $\mathbf{r}$ is the reset gate vector, $\mathbf{W}_{h,z,r}$, $\mathbf{U}_{h,z,r}$ are weight matrices, $\mathbf{b}_{h,z,r}$ are bias vectors, $\circ$ denotes element-wise vector multiplication and σ is the logistic sigmoid function. Similar equations hold for the backward GRU layer except that the $t-1$ index is replaced by $t+1$. The final output of the Bi-GRU layer is

$$\mathbf{y}_t = \overrightarrow{\mathbf{y}}_t + \overleftarrow{\mathbf{y}}_t \quad (6)$$

The BiGRU layer in this architecture contains 32 units.

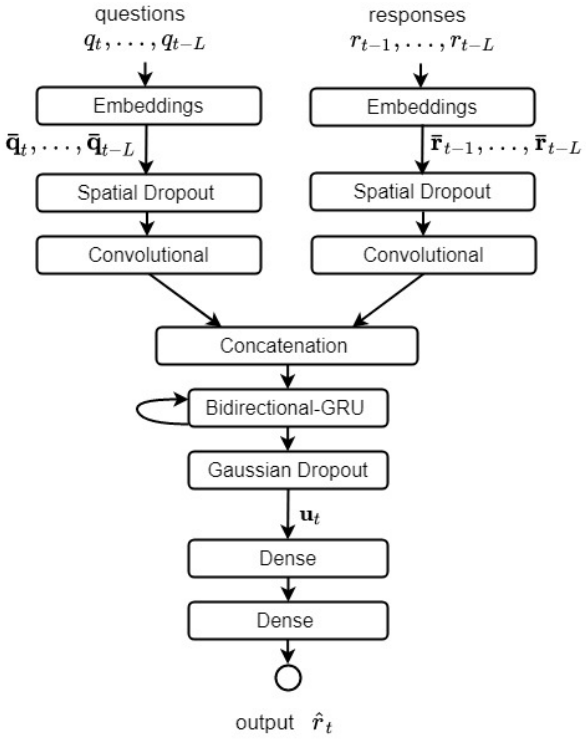


Fig. 4. The originally proposed Bi-GRU Model [16]

In our case, the input vector is the concatenation of the question and response embeddings filtered using 1-D convolutional operations:

$$\mathbf{x}_t = [\mathbf{F}_Q * \bar{\mathbf{q}}_{t-L:t}] \oplus [\mathbf{F}_R * \bar{\mathbf{r}}_{t-L:t-1}] \quad (7)$$

where the operation $*$ denotes 1-D convolution and $\oplus$ denotes concatenation. Regarding the experiments demonstrated in [16] the combination of an embeddings layer followed by a 1-D convolutional layer increases performance, presumably because it captures meaningful

local interactions in the input sequences. The Convolutional layer consists of 100 filters, with kernel size 3, stride 1, and ReLU activation function. Spatial dropout is used on the embedding prior to filtering in order to reduce overfitting. The value of the dropout percentage depends on the size of the examined dataset. The smaller the dataset size, the bigger the dropout percentage parameter value.

Depending on the values of the update and reset gates the Bi-GRU layer can have arbitrarily long memory. Therefore the output of the layer is a function of a potentially very long sequence of previous questions and responses:

$$\mathbf{u}_t \quad = \quad f(\bar{\mathbf{q}}_t, \bar{\mathbf{q}}_{t-1}, \bar{\mathbf{q}}_{t-2}, \ldots, \bar{\mathbf{r}}_{t-1}, \bar{\mathbf{r}}_{t-2}, \ldots) \quad (8)$$

The prediction of the correctness of the current answer r_t is a function of the current output of the dynamic unit:

$$\hat{r}_t \quad = \quad g(\mathbf{u}_t) \quad (9)$$

Since the target value r_t is binary, the function g corresponds to a classifier. This is implemented by a fully connected network which includes three dense layers with 50, 25 units with ReLU activation function and one output unit with sigmoid activation which used to make the final prediction $\hat{r}_t \in (0, 1)$. Note that Gaussian dropout [25] is applied to the output of the Bi-GRU layer before feeding the classification sub-network.

The overall layer structure of this early Bi-GRU model is depicted in Fig. 4.

B. Modified Dynamic KT-Bi-GRU Model

A drawback of the previous model is that the output $\mathbf{u}_t$ of the dynamic component cannot be easily associated with the knowledge state of the student at time t since it depends on the current question q_t for which we have no answer yet. Motivated by this observation, we propose an alternative KT-Bi-GRU model[2] depicted in Fig. 2b where the output of the dynamic component $\mathbf{v}_t$ is defined as a function of the previous student interactions (q_i, r_i), $i = t-1, t-2, \ldots$ *excluding* the current question q_t.

This is described by equation (10) where q_i and r_i are involved through their embeddings $\bar{\mathbf{q}}_i$ and $\bar{\mathbf{r}}_i$, respectively:

$$\mathbf{v}_t \quad = \quad f'(\bar{\mathbf{q}}_{t-1}, \bar{\mathbf{q}}_{t-2}, \ldots, \bar{\mathbf{r}}_{t-1}, \bar{\mathbf{r}}_{t-2}, \ldots) \quad (10)$$

In order to predict the correctness r_t of the student response to q_t, a new classifier function is required which combines $\mathbf{v}_t$ and $\bar{\mathbf{q}}_t$

$$\hat{r}_t \quad = \quad g'(\mathbf{v}_t, \bar{\mathbf{q}}_t) \quad (11)$$

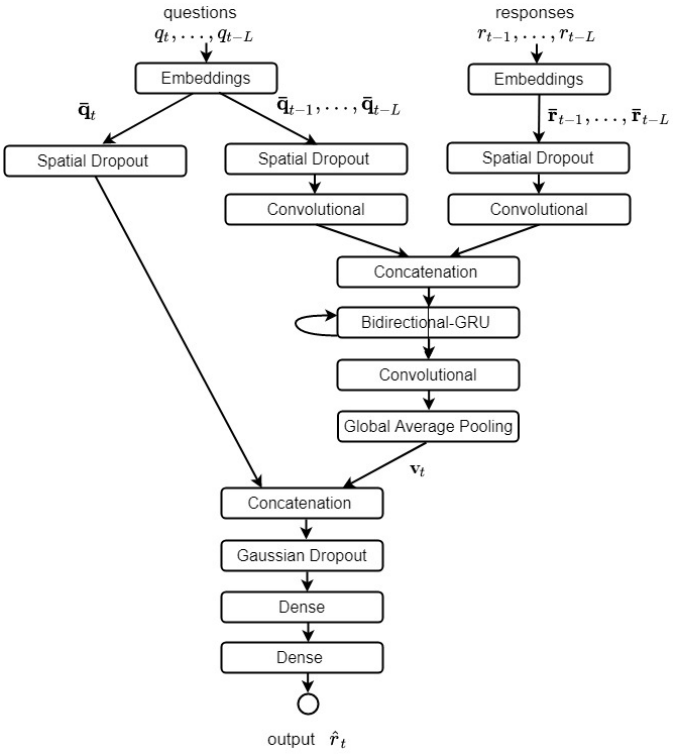


Fig. 5. The KT-Bi-GRU Model

Similar to our previous model, the KT-Bi-GRU model consists of three components: the Encoding, the Dy-

[2]GitHub KT-Bi-GRUlink will be provided in case the paper will be accepted

namic (BiGRU) component and the Classification component (Fig. 5b). The three components create two sub-networks: the Knowledge State Representation sub-net and Tracing sub-net for the student performance prediction. The common component of the both sub-nets is the Dynamic model. The Bi-GRU layer contains either 32 or 64 units. Batch normalization and the ReLU activation function are applied to the output of the Bi-GRU layer before generating $\mathbf{v}_t$. The classification component takes the input from two branches: the vector $\mathbf{v}_t$ with estimates the student's knowledge state and the representation vector $\bar{\mathbf{q}}_t$ of the current question. As with the previous model, the classifier contains three dense layers. The first two layers contain 50 and 25 units, respectively, with the ReLU activation function, while the final layer has 1 unit with the sigmoid function which predicts the student's response.

IV. DATASETS

To evaluate the models we used six publicly available benchmark datasets for the knowledge tracing task and generated a seventh one using the data of task 1 of the NeurIPS 2020 Educational Challenge [26]. All the datasets are related to the examination of mathematical problems. Three of the datasets were provided by the ASSISTments platform [27]. These are, the *ASSISTment09*, the *ASSISTment09 corrected*[3] and the *ASSISTment12*[4]. The fourth Assistements dataset, named *ASSISTment17*, was obtained from 2017 Data Mining competition page[5]. The fifth dataset, *FSAI-F1toF3* is provided by Find Solution Ai Limited and is collected using data from the 4LittleTrees[6] adaptive learning application. The sixth dataset *STATICS2011*[7] [28], is provided by a college-level engineering statics course. The seventh dataset called *NeurIPS-2020-small* was generated from the NeurIPS 2020 educational challenge[8] data provided by Eedi[9] team platform[10]. For each student in the original NeurIPS 2020 dataset a percentage of his/her interactions appear in the train set while the remaining interactions appear in the test set. However, in all other datasets the interactions of any student appear either in the train set or in the test set, but not in both. In order to comply with this paradigm, we created NeurIPS-2020-small as follows: (i) we combined the train and the test sets into a single dataset (ii) for practical reasons, due to the very large size of the original dataset, we kept only about 10% of the original data and removed the rows with missing values (iii) we split the dataset into train and test sets at a 70% / 30% ratio, in such a way that all student interactions appears exclusively either in the train set or in the test set. The description of all the datasets used in our experiments is shown in Table I.

In order to compare with previous works in the ASSISTment datasets we followed the common protocol where skill ids are used as inputs instead of questions in order to predict the student response. For the rest of the datasets we use question ids as inputs. Those questions, of course, are associated with one or more skills as explained above.

All of the above datasets, except for NeurIPS-2020-small, have been used to evaluate previous state-of-art knowledge tracing models including DKT, DKVMN and Deep-IRT. The datasets differ from each other in

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

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

[5]https://sites.google.com/view/assistmentsdatamining/data-mining-competition-2017

[6]https://www.4littletrees.com

[7]https://pslcdatashop.web.cmu.edu

[8]https://eedi.com/projects/neurips-education-challenge

[9]https://eedi.com/

[10]https://diagnosticquestions.com/

TABLE I
DATASETS OVERVIEW.

Dataset	Skills	Questions	Students	Responses
ASSISTment09	110	110*	4,151	325,637
ASSISTment09 corrected	101	101*	4,151	274,590
ASSISTment12	196	196*	28,834	2,036,080
ASSISTment17	101	101*	1,709	864,713
FSAI-F1toF3	99	2266$^+$	310	51,283
STATICS2011	98	1223$^+$	333	189,297
NeurIPS-2020-small	388	27570$^+$	7,733	1,696,690

*skills used as inputs to the prediction model.
+questions used as inputs to the prediction model.

the number of participating students, responses, skills, questions, and the number of records. We also used those benchmark datasets to evaluate the performance of all models in comparison to each other. In the first seven of the above data sets, there is a one-to-many relationship between skills and questions. In other words, each question corresponds to a single skill but the same skill may be associated with many questions. We refer to those datasets as the single-skill datasets.

In contrast, each question in NeurIPS-2020-small corresponds to a list of skills. At the same time, one skill is part of the list of skills in many questions. Thus NeurIPS-2020-small is referred to as a multi-skill dataset.

A. Data Pre-processing

Before proceeding with our evaluation experiments, we corrected grammatical errors in the skill names and replaced mathematical symbols with the corresponding words. For example, the word "*Polnomial*" in the skill name "*Parts of a Polnomial Terms Coefficient Monomial Exponent Variable*" was corrected to "*Polynomial*". Furthermore, the math symbols found in the skill names were converted to the corresponding words. For example, the symbols "*+,-,/,* ()*" in the skill name "*Order of Operations +,-,/,*() positive reals*" have been replaced with the words that express these symbols, ie. "*addition subtraction division multiplication parentheses*". This preprocessing action was preferred over the total removal of the math symbols since the skills refer to mathematical operations and deleting them, would have a destructive impact on the meaning of the skill. Moreover, the "*ASSISTments09 corrected*" and "*ASSISTments12*" datasets contained skills of the form of "*skill1_skill2*" and "*skill1_skill2_skill3*" which actually correspond to the same skill names. In this case we merged them into the first skill id, found before the first underscore symbol.

V. EXPERIMENTS

We experimentally compare the performance of our dynamic Bi-GRU models against the performance of the state-of-art knowledge tracing models DKT [5], DKVMN [6] and Deep-IRT [7] on the seven described datasets. Note that the python code [11] used for the DKT model experiments requires that the train/test split is performed during code execution, thus the data files have been converted to the appropriate format required for the experimentation process.

[11] https://github.com/lccasagrande/Deep-Knowledge-Tracing

A. Embedding vector initialization

The initialization of the response embeddings in our both Bi-GRU models, is done exclusively using random vectors. However, the questions are represented using embedding vectors which are initialized either randomly, or using pretrained vectors, based on the verbal description of the skill(s) corresponding to the questions. In our earlier model [16], for the pretrained initialization of the question embeddings we used the text files from Wikipedia2Vec[12] [29] that is based on Word2Vec method [30] and contains pretrainable embeddings for the word representation vectors in English language in 100 and 300 dimensions. For the same model we also used pretrained embeddings based on FastText [31] in 300 dimensions using the "*SISTER*" (SImple SenTence EmbeddeR)[13] library. In all the datasets each skill name consists of one or more words. Thus, the skill name embedding vector is created by adding the separate word embeddings in the skill name.

For the KT-Bi-GRU model, introduced in this work, we only apply the W2V method for skill names with initialization of vectors in 100 dimensions. We used the size of 100 dimensions based on the conclusion from the research on the our previous Bi-GRU model where we observed that the model performed better with this dimension size. The dimension 100 was used for the random initialization as well.

The pre-trained embedding vectors for each examined question in the NeurIPS-2020-small multi-skill dataset are generated based on the list of skills corresponding to the question. The process is completed in two steps:

- the representation vector of each skill is calculated by the W2V method as in the single skills data described above
- the question embedding vector representation is generated based on the average of the skills embedding vectors related to the question.

[12]https://wikipedia2vec.github.io/wikipedia2vec/

[13]https://pypi.org/project/sister/

B. Experimental Settings

We performed experiments with the aim to construct models with similar parameter settings for all the datasets we used. The hyperparameters that contribute to the best performance of KT-Bi-GRU model architecture are:

- L - window length of the student's interaction history. It is one of the main hyper-parameters that affects the inputs. We set $L = 50$ in for both Bi-GRU models since this value achieved the best results
- batch size: batches of data inputs, affects the speed of training
- embeddings initialization method: either random vectors or pre-trained embeddings
- question embeddings dimension and response embeddings dimension
- learning rate
- number and size of hidden layers: determines the complexity of the classification subnet
- recurrent units: concerns the number of training units of the Bi-GRU layer
- dropout rate
- epochs

During the experimental process, in addition to achieving the best performance, our goal was to adjust the hyperparameters with the greatest possible similarity for each data set in order to achieve the construction of a general model, suitable for data sets with different characteristics. The best parameter settings for the earlier Bi-GRU and the KT-Bi-GRU model are shown in Table II.

TABLE II
THE PARAMETER SETTINGS OF THE BOTH MODELS

Parameter	Early Bi-GRU	KT-Bi-GRU
L	50	50
batch size	50	100 or 300 (NeurIPS-2020-small)
skill/question embeddings initializing	random, W2V, FastText	random, W2V
responses initializing	random	random
embeddings vector dimension	100 & 300	100
learning rate	0.001	0.001 & 0.0001 (NeurIPS-2020-small)
hidden layers	2 layers with 50 and 25 units	2 layers with 50 and 25 units
recurrent units	32	64 & 32 (NeurIPS-2020-small)
dropout rate	0.2 or 0.9 due to the dataset	0.5-0.9 due to the dataset
training epochs	30	30

Both models have been trained using the cross-entropy loss and the Adam [32] optimization algorithm. Additionally, the learning rate was scheduled to begin from a starting value and decrease according to the epoch number n:

$$lr = \begin{cases} r_{init} & \text{if } n < 15 \\ r_{init} \times e^{(0.5 \cdot (15-n))} & \text{otherwise} \end{cases}$$

The parameter values in the new proposed model vary according to the data sets. The differences concern the parameters learning rate and recurrent units. For all the single-skill datasets the value of learning rate is 0.001 and the value of recurrent units is 64, while in "NeurIPS2020 small" multi-skill dataset the corresponding values of the parameters are 0.0001 and 32. Most variations concern dropout parameters values according to the number of skills or questions embeddings vectors and the size of database. In the case of ASSISTment datasets, the number of parameters to be trained by the neural network is less compared to other datasets. We consider this to be due to a combination of two characteristics, the input to the model during training is the representation of the skills and not the questions and the number of records in each dataset. To avoid overfitting, we regulated the values of the dropout hyper-parameters. So, after a thorough experimental procedure, depending on the input characteristics and the size of the datasets the dropout values for all the datasets were set as follows:

- ASSISTment 2009: spatial dropout = 0.6, gaussian dropout = 0.6,
- ASSISTment 2009 corrected: spatial dropout = 0.6, gaussian dropout = 0.6,
- ASSISTment 2012: spatial dropout = 0.5, gaussian dropout = 0.5,
- ASSISTment 2017: spatial dropout = 0.5, gaussian dropout = 0.5,
- FSAI-F1toF3: spatial dropout = 0.9, gaussian dropout = 0.8,
- STATICS 2011: spatial dropout = 0.8, gaussian dropout = 0.8,
- NeurIPS-2020-small: spatial dropout = 0.5, gaussian dropout = 0.5.

The evaluation metric that is used for the prediction of the probability correctness of student's response is Area Under the ROC Curve (AUC) [33]. This metric was used for the evaluation of all the state-of-art knowledge tracing models for the student performance prediction task.

VI. Results and Discussion

The experimental results are shown in Table III. As can be seen, the two dynamic Bi-GRU models, in general, outperform the previous state-of-art models in almost all datasets. One of the points we observed from the experimentation process is that the initialization of the skill embeddings, either randomly or by pretrained vectors representing the textual descriptions of the skills, does not significantly affect model performance. The addition of the spatial dropout layer between the embeddings and the convolutional layers helped achieve better performance. More details can be found at [16], the results of model performance on datasets are shown in Table III.

Comparing the recurrent Bi-GRU models with each other, we see that in the single-skill datasets the KT-Bi-GRU model performance is not far from our earlier model, while it is noticeably better in the case of the FSAIF-F1toF3 –the smaller of all the datasets– as well as in the case of NeurIPS-2020-small which is a multi-skill dataset. Moreover, we should note that the KT-Bi-GRU model has an advantage for tasks that require student knowledge estimation. For example, an important task is clustering students based on their knowledge state similarity [34], [35]. Our early Bi-GRU model is not suitable for this task since the internal representation vector $\mathbf{u}_t$ is not unique for a student with a specific history of question/response pairs since it also depends on the current question q_t. On the contrary, the output $\mathbf{v}_t$ of the recurrent layer in the KT-Bi-GRU model, is unique for a specific interaction sequence and therefore it can be used to represent the student knowledge state after he/she has completed this sequence. This is a key difference between the KT-Bi-GRU model compared to the other state-of-the-art models, as well. The exploitation of this novel feature of the proposed model is beyond the scope of the present paper which focuses on student performance prediction, however it deserves further investigation in the future.

The results also showed that in all single-skills datasets, the random question embeddings initialization method offers slightly better results. Therefore, the initial representation of the skills embeddings with the corresponding pretrained vectors did not contribute much to the model's performance. On the other hand, in the case of the NeurIPS-2020-small multi-skills dataset, the random question embedding initialization led to poor model training, with the test performance decreasing and after a few epochs stabilizing at a relatively low value (Figure 6a). Using the W2V question embedding initialization method, the model achieved an AUC value equal to 78.48 presented in Table III with the performance smoothly increasing in every epoch until convergence (Figure 6b). Thus, the questions were initially placed in the vector space in relation to the skills that concern them. According to the dataset's metadata, the skills are organized in a tree structure. There are skills that are branches of another skill, while a parent-skill can have more than one branch. Thus, the placement of the questions in the space based on the tree structure of the included skills contributed to the achievement of the model's better performance results.

Moreover, in order to have comparable results, we ran the codes of the previous models for all the datasets and found that only in the ASSISTment 2012 dataset, out of the seven data sets we used, we had lower results. The difference of the best performance of the known models in relation to the proposed knowledge tracing models is shown in Table IV. Also in the same table is shown the model that achieved better AUC results per dataset.

The performance of the DKT model could not be tested for NeurIPS data due to insufficient computing resources. While all the experiments for all the models

TABLE III
THE BEST RESULTS OF AUC METRIC PER DATASET AND PER MODEL (IN PERCENTS)

Models					
Dataset	DKT	DKVMN	Deep-IRT	Bi-GRU	KT-Bi-GRU
ASSISTment 2009	81,56	81,61	81,65	**82,55**	82,15
ASSISTment 2009 corrected	74,27	74,06	73,41	**75,27**	74,90
ASSISTment 2012	69,40	69,26	**69,73**	68,40	68,27
ASSISTment 2017	66,85	70,25	70,54	**73,76**	72,71
FSAI-F1toF3	69,42	68,40	68,69	70,47	**72,90**
STATICS 2011	82,71	83,17	83,09	**83,23**	82,88
NeurIPS 2020 small	-	75,14	74,78	77,85	**78,48**

TABLE IV
COMPARISON OF THE RANGE OF AUC RESULTS OF OUR AND PREVIOUS MODELS (IN PERCENTS)

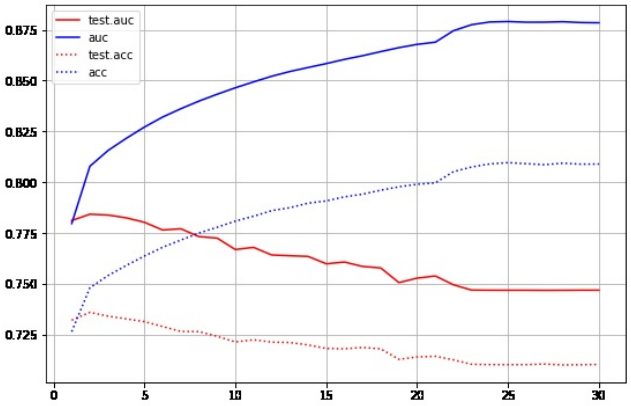
Dataset	KT-Bi-GRU vs Bi-GRU	KT-Bi-GRU vs Other previous models	The best model
ASSISTment09	-0,40	+0,50	Bi-GRU
ASSISTment09 corrected	-0,37	+0,63	Bi-GRU
ASSISTment12	-0,13	-1,46	Deep-IRT
ASSISTment17	-1,05	+2,17	Bi-GRU
FSAI-F1toF3	+2,37	+3,48	KT-Bi-GRU
STATICS2011	-0,35	-0,29	Bi-GRU
NeurIPS 2020 small	+0,63	+3,34	KT-Bi-GRU

and all the datasets were performed normally, in the DKT model with the NerIPS2020 small dataset there was a memory overload and inability to complete even the first training epoch of the model using the same code as in the rest of the datasets. One possible explanation for the lack of resources lies in the nature of the dataset itself. In contrast to all of the above datasets, NeurIPS records numerous responses per student and a very large number of different questions. Another possible explanation is that there is a bug in the code or that the model is not expected to support data with NeurIPS characteristics (a large number of different questions or a large number of answers from each user). Probably this volume of data can not be managed easily and we considered that there should be no intervention in the code of the DKT model that we borrowed for the research purpose.

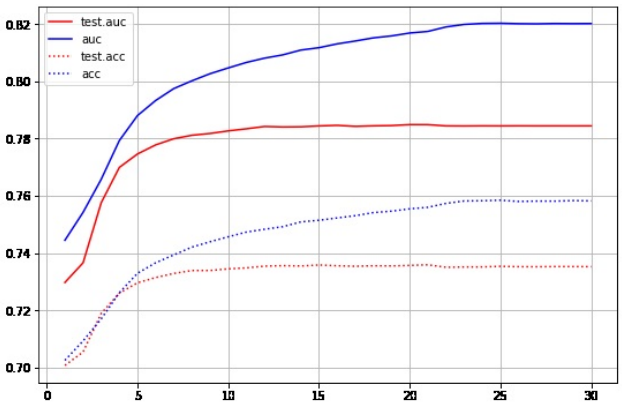
VII. CONCLUSION AND FUTURE WORK

In this paper we proposed a new recurrent Bidirectional GRU (KT-Bi-GRU) model for knowledge tracing and student performance prediction. The reference point of the proposed model is our earlier recurrent neural model, which surpassed the performance of the state-of-the-art models in most of the tested data sets. The KT-Bi-GRU model introduces a modified architecture with two sub-network parts. The first sub-network is for estimating the student knowledge state based on his/her interaction history using a recurrent neural network and the second sub-network predicts the student performance using multi-layer neural network.

The input data, ie. the student's interaction history, are encoded using embedding layers followed by 1-D convolutional layers. This layer combination was found



(a) Random question embeddings initialization



(b) W2V question embeddings initialization.

Fig. 6. Training/testing AUC and accuracy curves of the multi-skill NeurIPS2020-small dataset.

to improve performance. We also investigated different methods for initializing the question embedding layer with random vectors or pretrained embeddings. Our experiments showed that in the single-skill questions datasets there is no noticeable difference with respect to the initialization method used. On the contrary, in the NeurIPS dataset where a question involves multiple skills, the initialization using pretrained Word2Vec embeddings obtained from the verbal description of the skills contributed significantly to the performance improvement of the model.

Both of the Bi-GRU models are suitable either for single- or multi-skills datasets with the prospect of taking advantage of the fact that there is an assessment of the student's knowledge state. This information can play a key role in expanding our research to the production of personalized educational recommendations. In particular, in the KT-Bi-GRU model, where the estimation of the student knowledge state is based solely on the previous interaction history, the production of recommendations would not be affected by the current subject to be tested, which we consider more appropriate.

In the future we plan to exploit the proposed architecture by extending it to tasks that require student knowledge representation such as recommendation of educational content and student clustering. We also intend to investigate the effect of pretrained vector representations on data sets with different themes (other

than mathematics) and explore different initialization methods.

ACKNOWLEDGMENTS

We would like to thank NVIDIA Corporation for the kind donation of an Titan Xp GPU card that was used to run our experiments.

We acknowledge the valuable contribution of Mr. George Chrysogonidis and Mr. Vasileios Nikiforidis in the development of the original neural network model and the corresponding experimental evaluations.

REFERENCES

[1] A. T. Corbett and J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[2] Q. Liu, S. Shen, Z. Huang, E. Chen, and Y. Zheng, "A survey of knowledge tracing," *arXiv preprint arXiv:2105.15106*, 2021.

[3] H. Cen, K. Koedinger, and B. Junker, "Learning factors analysis–a general method for cognitive model evaluation and improvement," in *International Conference on Intelligent Tutoring Systems*. Springer, 2006, pp. 164–175.

[4] P. I. Pavlik Jr, H. Cen, and K. R. Koedinger, "Performance factors analysis–a new alternative to knowledge tracing." *Online Submission*, 2009.

[5] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *Advances in neural information processing systems*, vol. 28, pp. 505–513, 2015.

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

[7] C.-K. Yeung, "Deep-irt: Make deep learning based knowledge tracing explainable using item response theory," *arXiv:1904.11738*, 2019.

[8] T. Käser, S. Klingler, A. G. Schwing, and M. H. Gross, "Dynamic bayesian networks for student modeling," *IEEE Transactions on Learning Technologies*, vol. 10, pp. 450–462, 2017.

[9] S. Pu, G. Converse, and Y. Huang, "Deep performance factors analysis for knowledge tracing," in *International Conference on Artificial Intelligence in Education*. Springer, 2021, pp. 331–341.

[10] S. Yang, M. Zhu, J. Hou, and X. Lu, "Deep knowledge tracing with convolutions," *arXiv preprint arXiv:2008.01169*, 2020.

[11] W. Wang, T. Liu, L. Chang, T. Gu, and X. Zhao, "Convolutional recurrent neural networks for knowledge tracing," in *2020 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE, 2020, pp. 287–290.

[12] S. Shen, Q. Liu, E. Chen, H. Wu, Z. Huang, W. Zhao, Y. Su, H. Ma, and S. Wang, "Convolutional knowledge tracing: Modeling individualization in student learning process," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 1857–1860.

[13] R. Ma, L. Zhang, J. Li, B. Mei, Y. Ma, and H. Zhang, "Dtkt: An improved deep temporal convolutional network for knowledge tracing," in *2021 16th International Conference on Computer Science & Education (ICCSE)*. IEEE, 2021, pp. 794–799.

[14] Y. Liu, Y. Yang, X. Chen, J. Shen, H. Zhang, and Y. Yu, "Improving knowledge tracing via pre-training question embeddings," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 1577–1583.

[15] Z. He, W. Li, and Y. Yan, "Modeling knowledge proficiency using multi-hierarchical capsule graph neural network," *Applied Intelligence*, pp. 1–18, 2021.

[16] M. Delianidi, K. Diamantaras, G. Chrysogonidis, and V. Nikiforidis, "Student performance prediction using dynamic neural models," in *Fourteenth International Conference on Educational Data Mining (EDM 2021)*, 2021, pp. 46–54.

[17] A. Miller, A. Fisch, J. Dodge, A.-H. Karimi, A. Bordes, and J. Weston, "Key-value memory networks for directly reading documents," *arXiv preprint arXiv:1606.03126*, 2016.

[18] G. Abdelrahman and Q. Wang, "Knowledge tracing with sequential key-value memory networks," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 175–184, https://doi.org/10.1145/3331184.3331195.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, pp. 5998–6008, 2017.

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

[21] Y. Yang, J. Shen, Y. Qu, Y. Liu, K. Wang, Y. Zhu, W. Zhang, and Y. Yu, "Gikt: a graph-based interaction model for knowledge tracing," *arXiv preprint arXiv:2009.05991*, 2020.

[22] X. Song, J. Li, Y. Tang, T. Zhao, Y. Chen, and Z. Guan, "Jkt: A joint graph convolutional network based deep knowledge tracing," *Information Sciences*, vol. 580, pp. 510–523, 2021.

[23] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[24] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[26] Z. Wang, A. Lamb, E. Saveliev, P. Cameron, Y. Zaykov, J. M. Hernández-Lobato, R. E. Turner, R. G. Baraniuk, C. Barton, S. P. Jones, S. Woodhead, and C. Zhang, "Diagnostic questions: The neurips 2020 education challenge," *arXiv preprint arXiv:2007.12061*, 2020.

[27] AssistmentsData, "Assistments data," 2015, https://sites.google.com/site/assistmentsdata/.

[28] K. R. Koedinger, R. S. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper, "A data repository for the edm community: The pslc datashop," *Handbook of educational data mining*, vol. 43, pp. 43–56, 2010.

[29] I. Yamada, A. Asai, J. Sakuma, H. Shindo, H. Takeda, Y. Takefuji, and Y. Matsumoto, "Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020, pp. 23–30, https://wikipedia2vec.github.io/wikipedia2vec/.

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

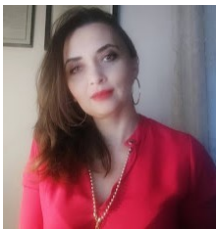
[31] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models," *arXiv preprint arXiv:1612.03651*, 2016.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] C. X. Ling, J. Huang, H. Zhang *et al.*, "Auc: a statistically consistent and more discriminating measure than accuracy," in *Ijcai*, vol. 3, 2003, pp. 519–524.

[34] N. A. Khayi and V. Rus, "Clustering students based on their prior knowledge." *International Educational Data Mining Society*, 2019.

[35] T. Omar, A. Alzahrani, M. Zohdy *et al.*, "Clustering approach for analyzing the student's efficiency and performance based on data," *Journal of Data Analysis and Information Processing*, vol. 8, no. 03, p. 171, 2020.



Marina Delianidi received the master's degree in web intelligence from the Alexandreio Technological Educational Institute of Thessaloniki, Greece, in 2015. She is currently working toward the Ph.D. degree with International Hellenic University, Thessaloniki, Greece. Her research focuses on knowledge tracing for the tasks of knowledge state estimation and performance prediction with the aim of recommending the proper educational content in order to improve the students' knowledge using machine learning methods.

Konstantinos Diamantaras (Senior Member, IEEE) received the Diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, and the Ph.D. degree from Princeton University, Princeton, NJ, USA, in 1992. He was a Postdoctoral Researcher with Siemens Corporate Research, Princeton, NJ, USA, and with the Aristotle University of Thessaloniki, Thessaloniki, Greece. In 1998, he joined the Department of Information Technology, the Technological Educational Institute of Thessaloniki, Thessaloniki, Greece. He is currently a Professor with the Department of Information and Electronic Engineering, International Hellenic University, Thessaloniki, Greece. His current research interests include machine learning, signal processing, wireless communications, and image processing. He was the Chairman to the Machine Learning for Signal Processing (MLSP) Technical Committee (TC) of the IEEE Signal Processing Society and was a Member of the MLSP and Signal Processing Theory and Methods TCs as well. He has served as Chairman and a Member of the TC for various machine learning, signal processing, and neural networks conferences. He was an Associate Editor for the IEEE Transactions on Signal Processing, IEEE Signal Processing Letters, and the IEEE Transactions on Neural Networks. He is currently an Associate Editor of the MDPI Journal of Applied Sciences.