Exploring Neural Network Models for the Classification of Students in Highly Interactive Environments

Tanja Käser
Graduate School of Education
Stanford University
tkaeser@stanford.edu

Daniel L. Schwartz
Graduate School of Education
Stanford University
daniel.schwartz@stanford.edu

ABSTRACT

Open-ended learning environments (OELEs) allow students to freely interact with the content and to discover important principles and concepts of the learning domain on their own. However, only some students possess the necessary skills for efficient and effective exploration. Guidance in the form of targeted interventions or feedback therefore has the potential to improve educational outcomes. A promising approach for adaptation in OELEs is the design of interventions based on the detection of characteristic learning behaviors through offline clustering, followed by a real-time classification of new students. In this paper, we explore the possibility of using recurrent neural network (RNN) models for this online classification task. We extensively evaluate the predictive performance of different variants of RNNs, namely long-short term memory models and gated recurrent units, and different architectures on a data set collected from an exploration-based educational game. We also compare the prediction accuracy of the different RNN models to the performance of traditional classifiers on the same data set. Our results demonstrate that RNNs perform similar or better than traditional methods regarding early classification and therefore constitute a promising alternative for the online classification of new students.

Keywords

recurrent neural networks, online classification, exploration environments, learning behavior

1. INTRODUCTION

Over the last years, there has been a rise in OELEs such as educational games or simulations. These environments allow students to freely interact with the content and (ideally) infer the concepts and principles of the learning domain through their exploration. However, previous research [20, 31, 24] has demonstrated that few students possess the problem solving and inquiry skills necessary to efficiently and effectively explore the space. Individualized guidance in the form of adaptive interventions or feedback therefore have the potential to improve students' exploration strategies and at the same time optimize the educational outcomes.

Traditionally, adaptation in computer-based learning environments has been based on the predictions of the student model. A large body of work has focused on developing student models that are able to accurately represent student knowledge. One of the most popular student modeling approaches is Bayesian Knowledge Tracing (BKT) [8], a technique that has been constantly refined and improved over the years, e.g., [34, 35]. Other widely used approaches are based on item response theory, such as the Additive Factors Model [5, 6] and Performance Factors Analysis [28]. Furthermore, dynamic Bayesian networks, e.g., [13, 18] have been used to model student knowledge. All of these approaches are based on the assumption that the knowledge of the student can be represented through a set of skills (knowledge components) and that we can infer the knowledge about a specific skill based on students' answers to tasks associated with this skill. OELEs do not fulfill these criteria as they (usually) do not provide specific sequences of tasks or explicitly define knowledge components. Therefore, the introduced student modeling techniques cannot be (directly) applied to such environments.

A prominent idea in the literature is to provide adaptation based on detected (and analyzed) learning behaviors. This idea has been formalized into a user modeling framework [15]: First, offline clustering is used to identify different types of student behaviors. The adaptive components of the environment are then designed with respect to the different behaviors found. Second, an online classification algorithm assigns new students to one of the clusters (and the corresponding intervention) in real time. A large amount of previous research has focused on the offline clustering part of the framework, applying clustering approaches to identify different types of learners [25, 3, 10]. [12] have represented student activity patterns in massive open online courses (MOOCs) using behavior state-transition graphs and demonstrated that the extracted patterns can be interpreted. Other work assessed students' problem solving behaviors in a game-based learning environment [32]. The full framework has been successfully applied to an environment for learning common artificial intelligence algorithms [15]. Other researchers [16] have used the framework to predict the mathematical learning patterns of students. Recently, the framework has been used to build student models for a more complex simulation of electric circuits [11]. To summarize, the presented research has mostly focused on offline clustering or the appli-

Tanja Käser and Daniel L. Schwartz "Exploring Neural Network Models for the Classification of Students in Highly Interactive Environments" In: *Proceedings of The 12th International Conference on Educational Data Mining (EDM 2019)*, Collin F. Lynch, Agathe Merceron, Michel Desmarais, & Roger Nkambou (eds.) 2019, pp. 109 - 118

cation of the framework to different learning domains, performing online classification using standard algorithms such as k-nearest neighbor [2].
Recurrent neural networks (RNN) have been successfully used for a variety of sequence classification problems such as sentiment analysis, e.g., [23] and video, e.g., [9]. RNNs have also been used in the educational community, for example to model student knowledge [30], to provide personalized recommendations in MOOCs [27], or to improve sensor-free affect detection [4]. Furthermore, RNNs have also been employed for classification problems. [26] suggested the use of long-short term memory (LSTM) models to classify learner behavior from touchscreen data. Other work [1] used LSTMs for the classification of problem-solving behaviors.

In this paper, we explore the use of RNNs for online classification: we assume the offline clustering solution to be given and train different classifiers to predict the cluster label of a new student early on during interaction with the OELE. We hypothesize that the ability of RNNs to handle sequences of arbitrary length, allowing them to accumulate the relevant information over each time step, may benefit the online classification task. We investigate different types of RNNs, varying the models along three dimensions: the internal node structure used (LSTMs and gated recurrent units (GRU)), the depth of the network (number of layers), and the number of nodes in the hidden layers. We also train the models to either predict the whole sequence, i.e., outputting the cluster label at each time step, or only predict the cluster label at the end of the sequence. The former approach has the advantage that the model is able to predict the cluster label at any point in time. The latter approach requires training different models to make predictions at specific time points, but enables the models to optimize predictions for the given point in time. We extensively evaluate and compare the predictive performance of all the different RNN models on a data set collected from an OELE [17]. Our results demonstrate that RNNs trained to predict the cluster label at each time step reach a similar predictive performance in early classification as the RNNs trained to predict the cluster label at the end of the sequence. Furthermore, despite the smaller number of parameters, GRU models tend to achieve a classification accuracy similar to LSTM models. We also compare the RNN models to traditional classifiers on the same data set. Our findings show that the RNN models perform similarly or better than the traditional approaches regarding the prediction of cluster labels early in the game. Earlier prediction of cluster labels allows to provide targeted guidance sooner. We therefore conclude that the use of RNNs for the online classification of student types is promising.

2. DATASET

The data set at hand was collected from a short interactive game aiming at assessing students' exploration choices.

Training Environment. `TugLet` is an interactive game designed to assess students' exploration behavior. The topic of the game revolves around a tug-of-war. Players can choose between two modes (illustrated in Fig. 1): they can engage in inquiry by simulating tug-of-war set-ups (*Explore*) or they can try to predict the winning side of specific tug-of-war set-ups and receive right-wrong feedback (*Challenge*). The

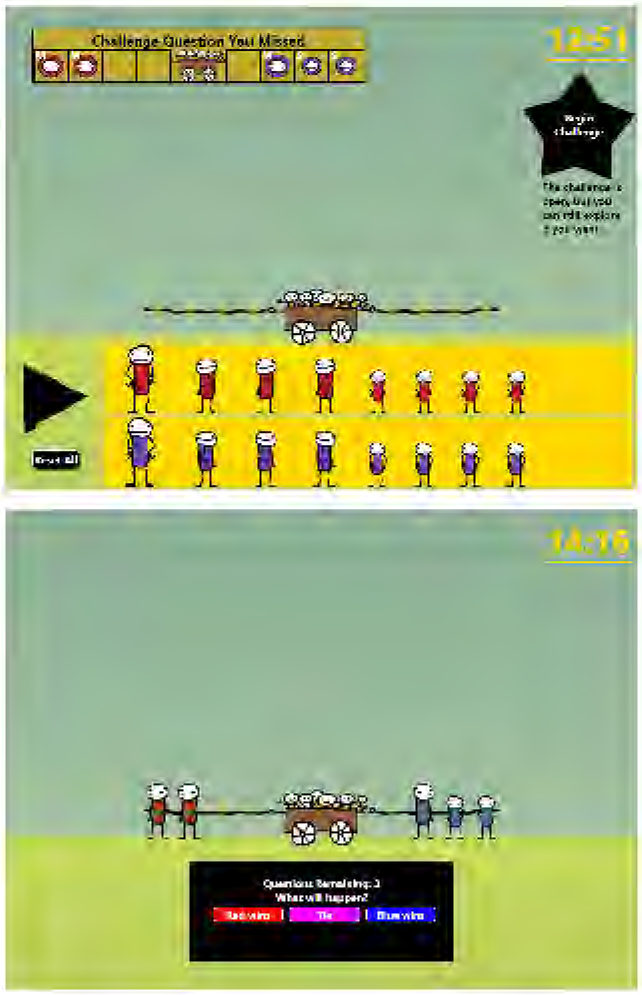

Figure 1: Snapshots of *Explore* (top) and *Challenge* modes (bottom). In *Explore* mode, children can set-up different tug-of-war teams and observe the outcome. In *Challenge* mode, children have to determine the winning side of specific tug-of-war set-ups.

Challenge mode consists of a maximum of eight problems ordered by increasing difficulty. The eight problems consist of one very easy question followed by two easy questions, two medium questions, and three difficult questions. If the student answers a problem incorrectly, (s)he is put back into *Explore* mode. The student is however free to choose to go back to the *Challenge* mode at any point in time. The game is over after players solve eight problems in a row correctly. The learning goal of the game, which is not revealed to the player, is to discover the mathematical principles underlying the tug-of-war.

Data Set. The data set consists of log files of 229 students attending the $8th$ grade of two different middle schools. The total number of observations in this data set is $10'258$. One observation corresponds to either one set-up tested in *Explore* mode or one question answered in *Challenge* mode. The length l of the observation sequences varies between $l = 12$ and $l = 127$. All students in this data set managed to pass the game.

Clustering Solution. Previous work [17] has shown that the learning outcome (measured by an external posttest) is not only influenced by students' exploration choices (*Explore* vs. *Challenge*) but also by the quality of their inquiry strategies. It was furthermore shown [19] that students can be grouped into six different clusters based on these detected strategies. These clusters can be sematically interpreted: cluster 1 captures students who systematically explore and try to understand the mathematical principles behind the tug-of-war. Cluster 3 consists of students who pass the game fast by only using *Challenge* mode. Students in cluster 6 also do not explore, but take a long time to pass the game. Students in cluster 4 on the other hand simulate many different tug-of-war configurations in *Explore* mode, without success. Cluster 2 lies in-between clusters 1 and 3, and exploration behaviors in cluster 5 are a mix between those in cluster 3 and cluster 6. The clusters are not only correlated to performance in an external posttest, but also predict academic achievement more broadly [19].

The features serving as an input for the clustering are extracted by level: children need to answer eight *Challenge* questions in a row correctly to pass the game and therefore the game can be divided into eight levels. Level n is reached the first time the student answers exactly n questions in a row correctly. The features used for clustering consist of the following cumulative counts extracted for level $n \in [1, 8]$: the number of *Challenge* problems NC_n needed to reach level n, the total number of tug-of-war set-ups NE_n simulated in *Explore* mode before reaching level n, and the number of tug-of-war set-ups NSE_n simulated in *Explore* mode which are classified as reflecting systematic inquiry (see [17] for a definition of systematic inquiry) until reaching level n. Therefore, the input features used for the clustering are $\mathbf{NC} = [NC_1, ..., NC_8]$, $\mathbf{NE} = [NE_1, ..., NE_8]$, and $\mathbf{NSE} = [NSE_1, ..., NSE_8]$. The cluster solution is then found by computing the pair-wise dissimilarities between all students for each feature using the Euclidean distance as a similarity measure and subsequently performing a pair-wise clustering [14]. The optimal number of clusters is determined by the Bayesian Information Criterion (BIC) [29]. In the following, we will us the presented clustering solution as ground truth for our classification task.

3. ONLINE CLASSIFICATION OF NEW STUDENTS

Ideally, the output of the clustering algorithm enables us to characterize different student behaviors and to design interventions or feedback based on the detected behaviors. Ideally, we are also able to characterize the performance of new learners in real time in order to deliver a targeted intervention as soon as possible. The corresponding framework is illustrated in Fig. 2. In the case of our data set, students assigned to cluster 4 might for example get hints on how to explore systematically, while students from cluster 3 will be prompted to use *Explore* mode in order to figure out the principles governing the tug-of-war.

RNN models are a family of neural network models able to handle sequences of arbitrary lengths. They are especially suited for time-series data and are able to represent the relevant information over a sequence of time steps. We therefore adapt different types of RNNs for the online classification task (marked in dark blue in Fig 2). RNNs map a sequence of input features $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_T}$ to a sequence of output features $\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_t}$. They maintain a sequence of hidden states $\mathbf{h_1}, \mathbf{h_2}, ..., \mathbf{h_t}$. These hidden states capture

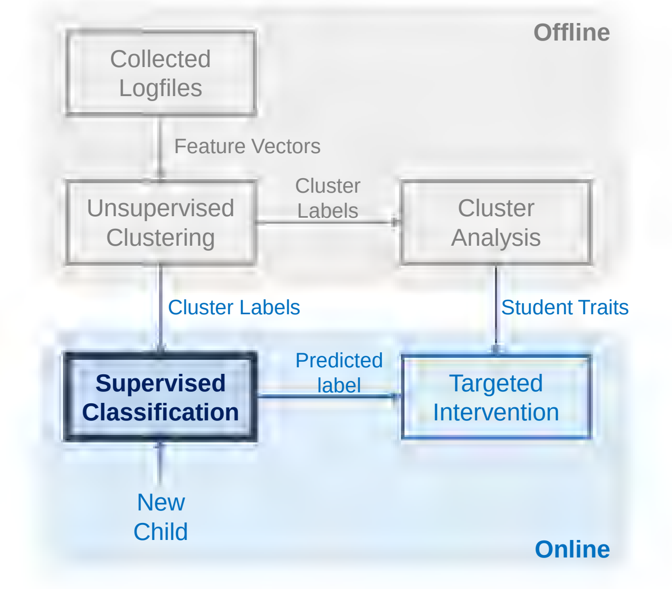

Figure 2: Students are clustered offline and targeted interventions are designed based on the (semantic) cluster interpretation. New students are then classified online. In this paper, we focus on the online classification task (marked in dark blue).

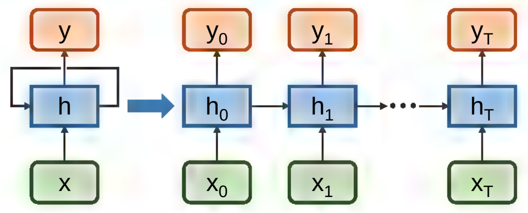

Figure 3: Simple RNN unrolled over T time steps. $\mathbf{x}$ denotes the input feature vectors, $\mathbf{y}$ denotes the output feature vectors and the hidden states are represented by $\mathbf{h}$.

relevant information from past observations which will influence future predictions. Figure 3 shows an illustration of a simple RNN model.

3.1 Long-Short Term Memory Classification

Long-short term memory (LSTM) models are a powerful modification of the RNN architecture.

Specification. LSTM models replace each hidden state $\mathbf{h_t}$ by an LSTM cell unit with additional gating parameters. These parameters determine when to forget or retain previous information. The update equations of an LSTM are as follows:

$$\mathbf{f_t} = \sigma(W_{fx}\mathbf{x_t} + W_{fh}\mathbf{h_{t-1}} + \mathbf{b_f}) \quad (1)$$

$$\mathbf{i_t} = \sigma(W_{ix}\mathbf{x_t} + W_{ih}\mathbf{h_{t-1}} + \mathbf{b_i}) \quad (2)$$

$$C'_t = \tanh(W_{Cx}\mathbf{x_t} + W_{Ch}\mathbf{h_{t-1}} + \mathbf{b_C}) \quad (3)$$

$$C_t = \mathbf{f_t} \times C_{t-1} + \mathbf{i_t} \times C'_t \quad (4)$$

$$\mathbf{o_t} = \sigma(W_{ox}\mathbf{x_t} + W_{oh}\mathbf{h_{t-1}} + \mathbf{b_o}) \quad (5)$$

$$\mathbf{h_t} = \mathbf{o_t} \times \tanh(C_t) \quad (6)$$

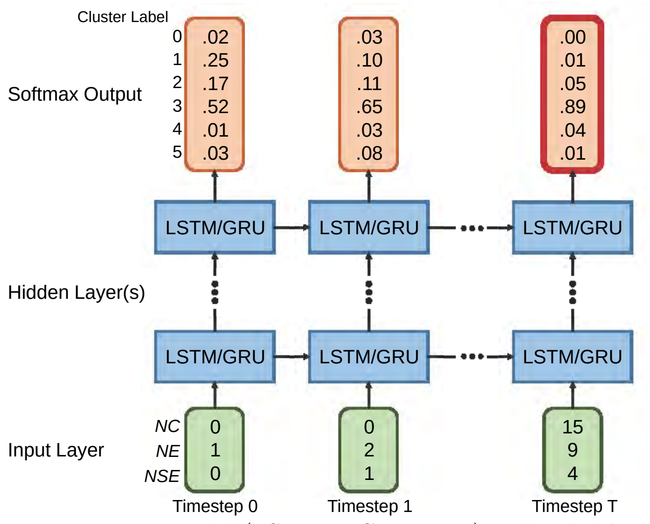

Figure 4: RNN model (LSTM or GRU units) with n hidden layers for an example student over T time steps. The input vector $\mathbf{x_t}$ contains the counts NC_t, NE_t, and NSE_t at each time step t. The output vector $\mathbf{\hat{y}_t}$ can be interpreted as a probability distribution over the different cluster labels.

Here, $\mathbf{f_t}$, $\mathbf{i_t}$, and $\mathbf{o_t}$ represent the forget, input, and output gates of the LSTM cell unit C_t. C'_t denotes an intermediate candidate cell state. The different weight matrices are described by W and the $\mathbf{b}$ stands for bias.

Modeling. For our task, the input vector $\mathbf{x_t}$ encodes the clustering features at each time step t. We input the counts for each time step t, i.e., $\mathbf{x_t} = [NC_t, NE_t, NSE_t]$. Let us assume that a hypothetical student m tested three different set-ups, the first two being random trials and the third one being systematic, followed by answering two *Challenge* questions. The input features for this student over the five described time steps are as follows: $\mathbf{x_{m,1}} = [0,1,0]$, $\mathbf{x_{m,2}} = [0,2,0]$, $\mathbf{x_{m,3}} = [0,3,1]$, $\mathbf{x_{m,4}} = [1,3,1]$, $\mathbf{x_{m,5}} = [2,3,1]$. Figure 4 details T time steps of an example student.
The output vectors $\mathbf{\hat{y}_t}$ represent the (predicted) cluster labels of the students: the output layer of the model uses the softmax function to normalize the vectors to sum to 1 such that the values within these output vectors can be thought of as probabilities for the different cluster labels (see Fig. 4). When training the model, we provide the cluster labels found during clustering as ground truth. Note that we use a one-hot encoding of the cluster labels, i.e. for a student m belonging to cluster $k = 2$, $\mathbf{y_{m,t}} = [0,0,1,0,0,0]$. The chosen model predicts the cluster label of the student at each time step t, augmenting the amount of available data and increasing flexibility, as it allows to predict the cluster label of a specific child at any point in time. We denote this type of model as LSTM_{Seq}.
Note that $\mathbf{y_{m,1}} = \mathbf{y_{m,2}} = ... = \mathbf{y_{m,T}}$ for all students m, because during clustering each student is assigned a fixed cluster label based on the whole sequence. Given that the cluster labels are fixed, we can also design the model to only output the cluster label at the end, i.e. train the LSTM to predict the cluster label at the end of a given input sequence. Instead of computing the training loss over the whole sequence, we calculate the loss only for the last output $\mathbf{\hat{y}_T}$ of the sequence (marked with red in Fig. 4). When using this type of model, we have to train a separate model for each prediction time point. In our case, we will train separate models for each level. We call this model LSTM_{End}.
Stacking multiple LSTM layers (see hidden layers in Fig. 4) is another possible variation of the architecture, i.e., the vector $\mathbf{h_t}$ of layer $n-1$ serves as an input for layer n. Stacking layers adds levels of abstraction of input observations over time, for example enabling representation of the problem at different time scales. We will denote LSTM models with n hidden layers, where $n > 1$, as $\text{LSTM}_{\text{Seq,n}}$ or $\text{LSTM}_{\text{End,n}}$.

3.2 Gated Recurrent Unit Classification

Gated recurrent unit (GRU) models are another powerful modification of RNN models. In contrast to LSTMs, they are less complex, making training more efficient.

Specification. Similar to LSTM models, GRU models replace each hidden unit $\mathbf{h_t}$ by a GRU cell unit with additional gating parameters. GRUs use update and reset gates, deciding what information should be passed to the output. The update equations of a GRU are as follows:

$$\mathbf{z_t} = \sigma(W_{fx}\mathbf{x_t} + W_{fh}\mathbf{h_{t-1}} + \mathbf{b_z}) \tag{7}$$
$$\mathbf{r_t} = \sigma(W_{ix}\mathbf{x_t} + W_{ih}\mathbf{h_{t-1}} + \mathbf{b_i}) \tag{8}$$
$$\mathbf{h'_t} = \tanh(W_{h'x}\mathbf{x_t} + r_t \times W_{h'h}\mathbf{h_{t-1}} + \mathbf{b'_h}) \tag{9}$$
$$\mathbf{h_t} = \mathbf{z_t} \times \mathbf{h_{t-1}} + (1 - \mathbf{z_t}) \times \mathbf{h'_t} \tag{10}$$

$\mathbf{z_t}$ and $\mathbf{r_t}$ represent the update and reset gate of the GRU cell unit. h'_t denotes an intermediate candidate hidden state. The different weight matrices are described by W and the $\mathbf{b}$ stands for bias.

Modeling. Just as for the LSTM models, the clustering features at each time step t are represented by the input vector $\mathbf{x_t}$, i.e., $\mathbf{x_t} = [NC_t, NE_t, NSE_t]$. The input sequence of an example student is given in Fig. 4. We also use the exact same description for the output layer of the GRU: the output layer of the model uses the softmax function to normalize the vectors to sum to 1 such that the values within the output vectors $\mathbf{\hat{y}_t}$ can be interpreted as probabilities for the different cluster labels (see Fig. 4). We again train a model on the whole output sequences of the students able to predict the cluster label at each time step t. We denote this model with GRU_{Seq}. We also train one GRU model per level, where we compute the loss of the model only for the last output $\mathbf{\hat{y}_T}$ of the sequence (marked with red in Fig. 4). We denote this model with GRU_{End}. Finally, similar to LSTM models, we can also stack GRU models on top of each other. We will call models with n hidden layers, where $n > 1$, $GRU_{\text{Seq,n}}$ or $GRU_{\text{End,n}}$.

4. EXPERIMENTAL EVALUATION

We evaluated the predictive accuracy of the variations of RNN models on the data set described in Section 2. We also compared them to the following popular traditional classifiers using the same data set: k-Nearest-Neighbor (kNN) and random forests (RF). While these classifiers are well tested and efficient to train, they require features being the same length for each student and therefore need to be trained for fixed points in time. RNNs on the other hand represent the relevant information over time, possibly enabling a more accurate classification of students early in the game.

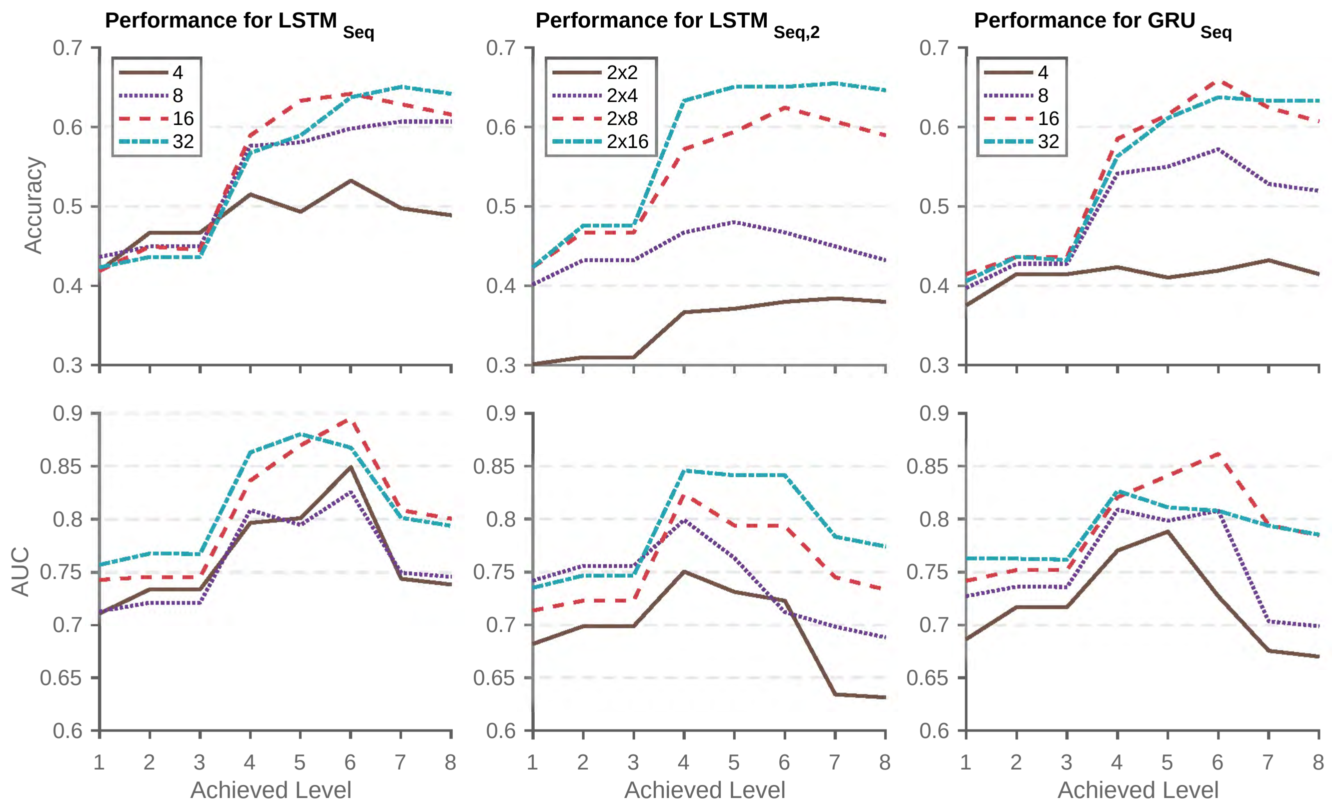

Figure 5: Accuracy (top) and AUC (bottom) of the LSTM_{Seq}, $\text{LSTM}_{\text{Seq},2}$, and GRU_{Seq} models by achieved level. Both measures increase up to level 6 and stagnate or deteriorate afterward. This is probably due to the fact that the models are trained to predict the cluster label after each time step t, i.e. training loss is optimized over the whole sequence of observations.

Experimental Setup. We applied a train-test setting, i.e. parameters were fit on the training data set and performance of the methods was evaluated on the test data set. Predictive performance was evaluated using the accuracy as well as the micro-averaged area under the ROC curve (AUC). The accuracy is a measure that can be interpreted easily. The cluster solutions for both data sets are not balanced. We used the AUC as an additional performance measure as it is robust to class imbalance.

For all methods, we used a student-stratified (i.e. dividing the folds by students) 10-fold cross validation. Within each fold f, we re-clustered the students of the training data set of f to obtain the output features $\mathbf{y}$, i.e. the cluster labels, for training. We purposely did not use the original cluster labels from the solution found on the whole data set (including training and test data) for training, to prevent dependencies to the cluster labels of the test data set. The average cluster stability [22] between the 10 different training data sets and the original cluster solution was 0.87, i.e., on average 87% of the samples received the same label on the training data set as on the original data set. Therefore, 0.87 constitutes an upper bound for the accuracy of the classifiers.

All the RNN models were implemented using Keras [7] with Theano [33] as backend. Categorical crossentropy was used to calculate loss and ADAM was used as the optimizer. The models were trained for $e = 100$ epochs. For all types of RNN models, we used post-padding and masking to account for the different sequence lengths.

For the traditional classifiers, we trained one model for each level n of the game. The input vector $\mathbf{x_n}$ of each model therefore encodes the clustering features exactly at level n, i.e., $\mathbf{x_n} = [NC_n, NE_n, NSE_n]$.

We determined the optimal number k_o of nearest neighbors for the kNN classifier as follows: within each fold f, we randomly put 10% of the students from the training data set in a validation data set and selected the number of nearest neighbors $k_{o,f}$ yielding the best performance in terms of accuracy on this validation data set. We then predicted the cluster labels of the test data set using the labels of the $k_{o,f}$ nearest neighbors.

For the RF method, we trained $B = 100$ binary decision trees using bootstrapping with re-sampling ($r_f = 1.5 \cdot M_f$, with M_f being the number of samples in the training data set of fold f).

RNN Models returning a sequence of outputs. We varied the parameters of our RNN models outputting the whole sequence of cluster labels along three dimensions: the structure of the hidden layer(s) (LSTM or GRU), the number of hidden layers, and the dimension of the hidden state within a hidden layer. Specifically, we computed the predictive performance for the following models: a 1-layer LSTM (LSTM_{Seq}) with a d_h-dimensional hidden state where $d_h \in \{4, 8, 16, 32\}$, a 2-layer LSTM ($\text{LSTM}_{\text{Seq},2}$) with a d_h-dimensional hidden state per layer where $d_h \in \{2, 4, 8, 16\}$, and a 1-layer GRU (GRU_{Seq}) with a d_h-dimensional hidden state where $d_h \in \{4, 8, 16, 32\}$. Figure 5 illustrates the predictive performance in terms of accuracy and AUC for the three

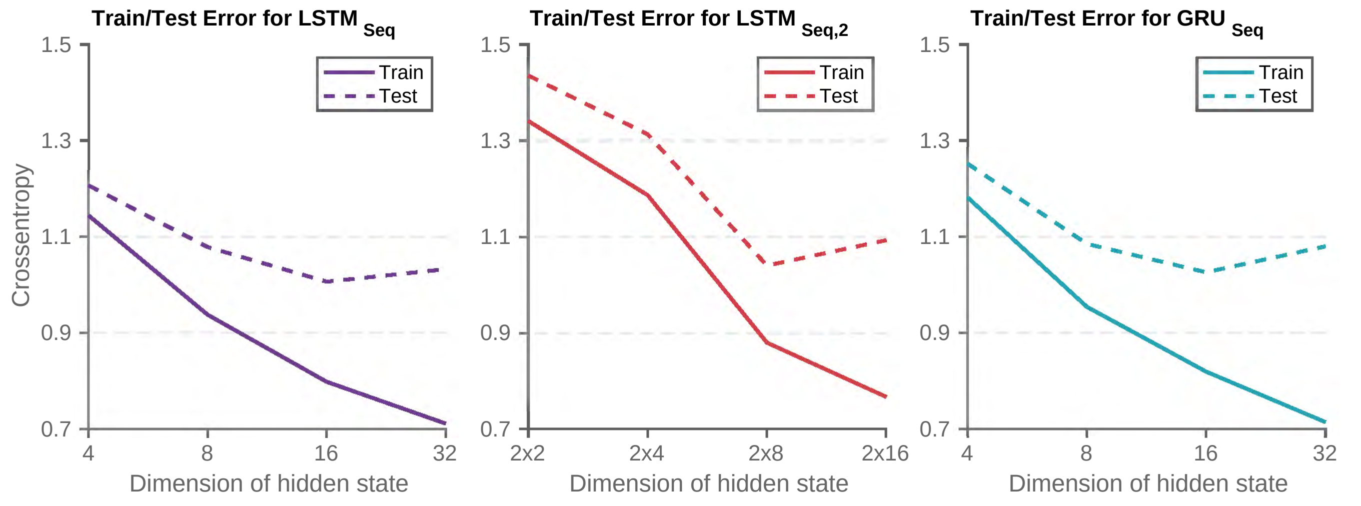

Figure 6: Categorical crossentropy of the $\mathrm{LSTM}_{\mathrm{Seq}}$, $\mathrm{LSTM}_{\mathrm{Seq},2}$, and $\mathrm{GRU}_{\mathrm{Seq}}$ models by the number of dimensions of the hidden state. The average test error begins to increase when using more than $d_h = 16$ hidden dimensions, while the training error still decreases.

different architectures by achieved level.

The $\mathrm{LSTM}_{\mathrm{Seq}}$ models reach a poor accuracy when $d_h = 4$. There is also not much difference between the accuracy at level 1 (0.42) and at level 8 (0.49). The accuracy improves substantially when increasing the dimension of the hidden state to $d_h = 8$, $d_h = 16$, or $d_h = 32$. We especially observe a jump in accuracy between levels 3 (e.g., $\mathrm{Accuracy}_8 = 0.45$) and 4 (e.g., $\mathrm{Accuracy}_8 = 0.58$). For $d_h = 32$, there is a second jump between levels 5 ($\mathrm{Accuracy}_{32} = 0.59$) and 6 ($\mathrm{Accuracy}_{32} = 0.64$). These jumps in accuracy correspond to jumps in the difficulty of the *Challenge* questions: the eight problems consist of one very easy question followed by two easy questions, two medium questions, and three difficult questions. There is no increase in accuracy between levels 1 and 3 as most students passed these easy levels very quickly. We observe a similar picture for the AUC. The AUC increases with the number of dimensions d_h of the hidden state. Note that the AUC again jumps between levels 3 (e.g., $\mathrm{AUC}_8 = 0.72$) and 4 (e.g., $\mathrm{AUC}_8 = 0.81$).

For the $\mathrm{LSTM}_{\mathrm{Seq},2}$ models, predictive performance again increases with the number of dimensions of the hidden state within a layer. For this type of models, using a 2-dimensional hidden state or a 4-dimensional hidden state per layer leads to a low accuracy. Increasing to an 8-dimensional hidden state or a 16-dimensional hidden state per layer yields a large improvement in accuracy and these models also show a jump in accuracy between level 3 (e.g., $\mathrm{Accuracy}_{2x16} = 0.48$) and level 4 (e.g., $\mathrm{Accuracy}_{2x16} = 0.63$). The AUC also increases with an increasing number of dimensions d_h per hidden layer. The $\mathrm{LSTM}_{\mathrm{Seq},2}$ models' accuracy is in range with the accuracy of the $\mathrm{LSTM}_{\mathrm{Seq}}$ models for $d_h = 16/d_h = 2x8$ hidden dimensions as well as for $d_h = 32/d_h = 2x16$ hidden dimensions. However, the $\mathrm{LSTM}_{\mathrm{Seq}}$ models show a higher AUC than the $\mathrm{LSTM}_{\mathrm{Seq},2}$ models, e.g., for $d_h = 16/d_h = 2x8$ hidden dimensions (for example at level 5: $\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{Seq}}} = 0.87$, $\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{Seq},2}} = 0.79$).

Performance of the $\mathrm{GRU}_{\mathrm{Seq}}$ models in terms of accuracy shows the same trends over time as for the $\mathrm{LSTM}_{\mathrm{Seq}}$ models. Again, employing a 4-dimensional hidden state results in a low accuracy. When increasing the number of dimensions of the hidden state to $d_h = 8$, $d_h = 16$, or $d_h = 32$, the models are able to capture the jump in difficulty between level 3 (e.g., $\mathrm{Accuracy}_8 = 0.43$) and level 4 (e.g., $\mathrm{Accuracy}_8 = 0.54$). The architecture employing a 16-dimensional hidden state also shows a jump in accuracy between level 4 ($\mathrm{Accuracy}_{16} = 0.59$) and level 6 ($\mathrm{Accuracy}_{16} = 0.66$). The AUC again increases with an increasing number of dimensions of the hidden state. Applying $d_h = 32$ instead of $d_h = 16$ hidden dimensions does generally not increase the AUC, and is even worse for some levels, e.g., for level 6 ($\mathrm{AUC}_{16} = 0.86$, $\mathrm{AUC}_{32} = 0.81$). Generally performance is in range with the performance of the $\mathrm{LSTM}_{\mathrm{Seq}}$ models in terms of accuracy for $d_h = 16$ and $d_h = 32$ hidden dimensions. Again, the AUC for the $\mathrm{LSTM}_{\mathrm{Seq}}$ models tends to be higher than the AUC for the $\mathrm{GRU}_{\mathrm{Seq}}$ models, for example for $d_h = 16$ hidden dimensions at the peak level 6 ($\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{Seq}}} = 0.90$, $\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{GRU}}} = 0.86$).

The performance increase of all models with a higher number of hidden dimensions is as expected. However, the danger of overfitting increases with a higher number of parameters. While our training and evaluation methods have measures for overfitting, such as the crossvalidation, in place, we investigated the relation between the average training and test error of the different configurations and the number of dimensions d_h of the hidden state. Figure 6 illustrates the average categorical crossentropy on the training data sets and test data sets. We observe that the difference between training error and test error starts to get bigger with an increased number of hidden dimensions. Specifically, there is a kink in the test error at $d_h = 16/d_h = 2x8$ hidden dimensions for all models. We therefore conclude that $d_h = 16/d_h = 2x8$ is the maximum number of hidden dimensions that should be used for this data set.

RNN Models returning the last output in the sequence. We also trained a range of RNN models returning

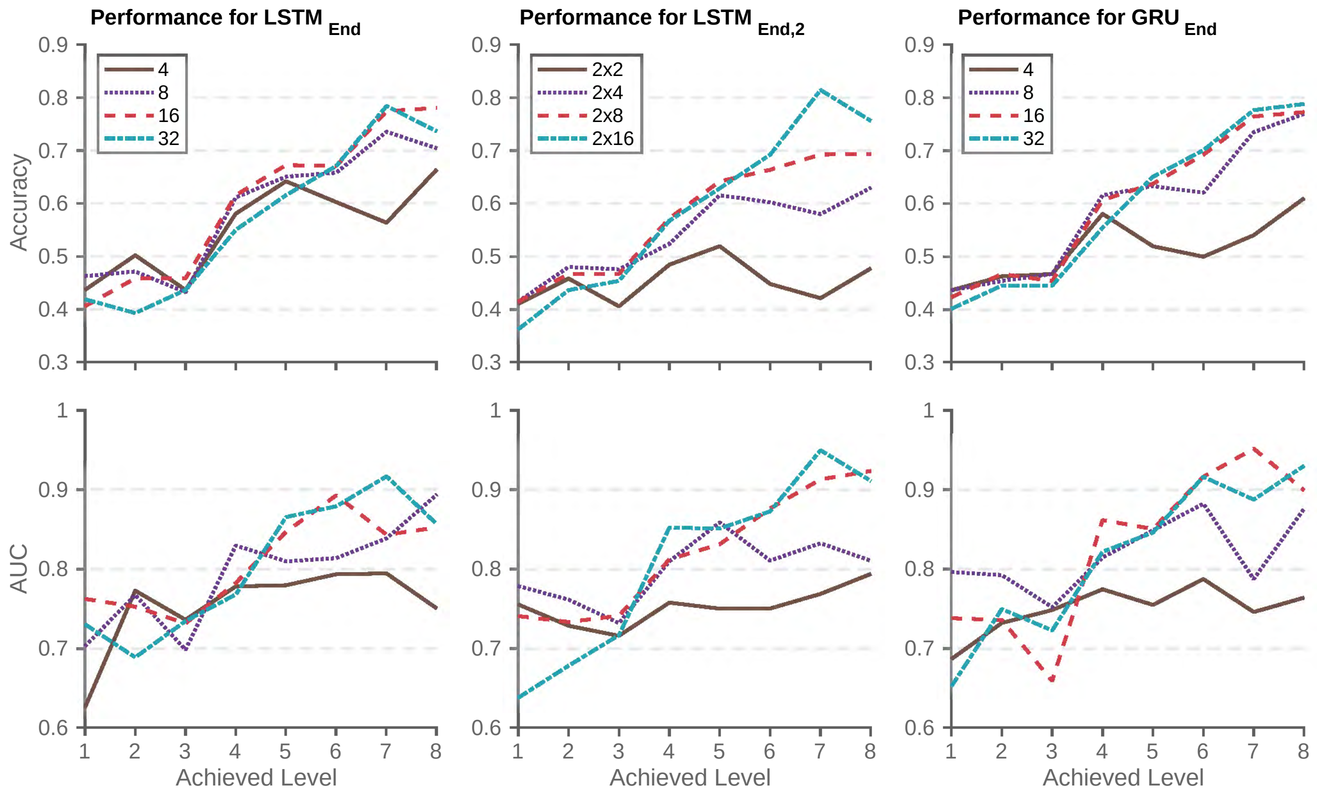

Figure 7: Accuracy (top) and AUC (bottom) of the LSTM_{End}, $\text{LSTM}_{\text{End,2}}$, and GRU_{End} models by achieved level. Both measures increase over time and all models achieve a similar predictive performance for the higher numbers of hidden dimensions.

only the last output in the sequence, i.e. predicting the cluster label at the end of a given sequence. We varied the parameters of these models along the same dimensions as for the models predicting the whole sequence. Specifically, we computed the predictive performance for the following models: a 1-layer LSTM (LSTM_{End}) with a d_h-dimensional hidden state where $d_h \in \{4, 8, 16, 32\}$, a 2-layer LSTM ($\text{LSTM}_{\text{End,2}}$) with a d_h-dimensional hidden state per layer where $d_h \in \{2, 4, 8, 16\}$, and a 1-layer GRU (GRU_{End}) with a d_h-dimensional hidden state where $d_h \in \{4, 8, 16, 32\}$. We trained one model for each level of the game. The predictive performance of all the models in terms of accuracy and AUC is illustrated in Fig. 7.

Similar to the models predicting sequences, the LSTM_{End} model achieves the lowest accuracy with $d_h = 4$. However, up to level 6, there is no big difference in accuracy between this model and the models with $d_h \geq 8$ hidden dimensions. All the LSTM_{End} models capture the jump in difficulty between level 3 and level 4. The models with higher-dimensional hidden states also capture the second jump happening after level 5. For $d_h = 16$, we for example see a jump in accuracy after level 3 (level 3: Accuracy $= 0.46$, level 4: Accuracy $= 0.62$) and a second jump after level 6 (level 6: Accuracy $= 0.67$, level 7: Accuracy $= 0.77$). Regarding the AUC, we observe superior performance of the models with $d_h = 16$ and $d_h = 32$ hidden dimensions after level 4. For these models, the AUC constantly increases until level 6 ($\text{AUC}_{16} = 0.89$, $\text{AUC}_{32} = 0.88$). As seen before, we do not observe any improvement in performance after level 6.

For the $\text{LSTM}_{\text{End,2}}$ model, employing $d_h = 2$ hidden dimensions per layer leads to the lowest achieved accuracy and there is also not much improvement over time. The models with $d_h > 2$ capture the increased difficulty after level 3 (e.g., level 3: $\text{Accuracy}_{2x8} = 0.47$, level 4: $\text{Accuracy}_{2x8} = 0.57$). Only the models using $d_h = 8$ or $d_h = 16$ hidden dimensions per layer manage to capture the second jump in difficulty (e.g., level 6: $\text{Accuracy}_{2x16} = 0.69$, level 7: $\text{Accuracy}_{2x16} = 0.81$). We observe a similar picture for the AUC: when using using $d_h = 8$ or $d_h = 16$ hidden dimensions, there is a strong increase in AUC after level 3 (e.g., level 3: $\text{AUC}_{2x8} = 0.74$, level 4: $\text{AUC}_{2x8} = 0.81$) and after level 6 (e.g., level 6: $\text{AUC}_{2x16} = 0.87$, level 7: $\text{AUC}_{2x16} = 0.95$).

The accuracy plot of the GRU_{End} models looks similar to the accuracy plot of the LSTM_{End} models. Up to level 4, all models perform similarly (at level 4: $\text{Accuracy}_4 = 0.58$, $\text{Accuracy}_8 = 0.62$, $\text{Accuracy}_{16} = 0.61$, $\text{Accuracy}_{32} = 0.55$). For the higher levels, the model with $d_h = 4$ hidden dimensions shows the lowest accuracy. Using a model with an 8-dimensional hidden state nicely captures the jumps in accuracy between level 3 (Accuracy $= 0.47$) and level 4 (Accuracy $= 0.62$) and between level 6 (Accuracy $= 0.62$) and level 7 (Accuracy $= 0.73$). Increasing the number of hidden dimensions to $d_h = 16$ improves performance only from level 5 on (e.g., at level 6: $\text{Accuracy}_8 = 0.62$, $\text{Accuracy}_{16} = 0.69$). This model also captures the two jumps in accuracy. Using $d_h = 32$ hidden dimensions does not lead to any further improvements. The model employing a 4-dimensional hidden state performs worst for the AUC, with exception of level 3. When using $d_h = 16$ or $d_h = 32$ hidden dimen-

sions, the AUC again models the two jumps in difficulty (e.g., level 3: $\mathrm{AUC}_{32} = 0.72$, level 4: $\mathrm{AUC}_{32} = 0.82$, level 5: $\mathrm{AUC}_{32} = 0.85$, level 6: $\mathrm{AUC}_{32} = 0.92$). It also seems that using a higher number of hidden dimensions, i.e. $d_h > 8$, increases the stability of the AUC. With exception of the drop at level 3, the AUC of the model with $d_h = 16$ hidden dimensions nicely increases over time.
We again tested for overfitting, by comparing the average training and test loss of the different models. Just as for the models trained to predict the cluster label at each time step, we found that there is a kink at $d_h = 16$ hidden dimensions: while the training error still decreases for a higher number of d_h, the error on the test set increases. We therefore conclude that $d_h = 16$ is the maximum number of hidden dimensions that can be used.

'Sequence versus End'. When comparing the RNN models trained for sequence prediction to the models trained to predict only the last output of the sequence, we observe that the performance plots show the same overall trends (see Fig. 5 and Fig. 7). For both types, predictive performance in terms of the accuracy is similar for the 1-layer LSTM and the 1-layer GRU models. The $\mathrm{GRU}_{\mathrm{Seq}}$ model generally has a lower accuracy than the $\mathrm{LSTM}_{\mathrm{Seq}}$ model when using $d_h < 16$. The $\mathrm{GRU}_{\mathrm{End}}$ model exhibits a lower accuracy than the $\mathrm{LSTM}_{\mathrm{End}}$ model only for $d_h = 4$. The models with two stacked layers, i.e. the 2-layer LSTM models, generally show a lower accuracy when employing a low number of hidden dimensions per layer ($2x2$ or $2x4$).
For both the 'sequence' and the 'end' RNN models, all three model types achieve similar accuracies for $d_h = 8$ or $d_h = 16$ hidden dimensions. All RNN models show no improvement or even a drop in AUC after level 6. This effect is more pronounced for the models which are trained on the sequence, as their loss is optimized over the whole sequence. We further hypothesize that the length of the sequences at the higher levels might be too long for the RNN models to capture the relevant information, because even with the LSTM architecture, RNNs tend to struggle with very long data sequences. The main difference between the 'sequence' and the 'end' model is the larger increase of the accuracy with increasing levels. For example, for the $\mathrm{LSTM}_{\mathrm{Seq}}$ with $d_h = 16$ the accuracy at level 1 is 0.42 and the accuracy at level 7 is 0.63, while the accuracy of the $\mathrm{LSTM}_{\mathrm{End}}$ model with $d_h = 16$ is 0.41 at level 1 and 0.77 at level 7. Because the 'end' models are optimized to predict the last output of a sequence, they reach a higher accuracy at the end of the game (e.g. at level 8: $\mathrm{Accuracy}_{\mathrm{GRU}_{\mathrm{Seq}},16} = 0.61$, $\mathrm{Accuracy}_{\mathrm{GRU}_{\mathrm{End}},16} = 0.77$). The 'sequence' RNN models are optimal on average and therefore exhibit less variance over time and smoother accuracy and AUC curves. For the medium levels of the game, 'sequence' and 'end' RNN models perform similar in terms of accuracy and AUC.
All configurations exhibit a satisfying accuracy and a medium-high AUC for $d_h \geq 8$. As a comparison, a random classifier would achieve an accuracy of 0.17 and the accuracy of a classifier always predicting the majority label would be 0.32. The AUC of these two classifiers would be 0.5.

Comparison to traditional classifiers. We compared the predictive performance of selected RNN models to the predictive performance of the traditional classifiers. Because the RNN models show an increased performance with a higher number of hidden dimensions, we used RNN models with the maximum number of $d_h = 16$ not resulting in overfitting for comparison. In case of the 'sequence' models, the $\mathrm{LSTM}_{\mathrm{Seq}}$ model achieved a similar accuracy, but a higher AUC than the two other model types for $d_h = 16$. We therefore selected the $\mathrm{LSTM}_{\mathrm{Seq}}$ model with $d_h = 16$ for comparison. In case of the 'end' models, the 1-layer LSTM and 1-layer GRU models achieved similar results, however, the LSTM models were better at predicting the jumps in difficulty. We therefore selected the $\mathrm{LSTM}_{\mathrm{End}}$ model with $d_h = 16$ for comparison. Figure 8 displays the accuracy and the AUC of the kNN classifier, the RF method, a 1-layer LSTM model for sequence predicting with $d_h = 16$ hidden dimensions ($\mathrm{LSTM}_{\mathrm{Seq}}$), and a 1-layer LSTM model for predicting the last output of the sequence with $d_h = 16$ hidden dimensions ($\mathrm{LSTM}_{\mathrm{End}}$).
We observe that the RNN models outperform the traditional classifiers for the first three levels regarding the accuracy (e.g., at level 3: $\mathrm{Accuracy}_{\mathrm{kNN}} = 0.38$, $\mathrm{Accuracy}_{\mathrm{RF}} = 0.43$, $\mathrm{Accuracy}_{\mathrm{LSTM}_{\mathrm{Seq}}} = 0.45$, $\mathrm{Accuracy}_{\mathrm{LSTM}_{\mathrm{End}}} = 0.46$). The same holds for the middle of the game, i.e. levels $4-6$ (e.g., at level 5: $\mathrm{Accuracy}_{\mathrm{kNN}} = 0.57$, $\mathrm{Accuracy}_{\mathrm{RF}} = 0.63$, $\mathrm{Accuracy}_{\mathrm{LSTM}_{\mathrm{Seq}}} = 0.63$, Accu- $\mathrm{racy}_{\mathrm{LSTM}_{\mathrm{End}}} = 0.67$). At the end of the game, the accuracy of the traditional classifiers is close to the stability of the clustering ($\mathrm{Accuracy}_{\mathrm{kNN}} = 0.85$, $\mathrm{Accuracy}_{\mathrm{RF}} = 0.83$).
For the RF approach and the two RNN models, we also computed the AUC (see Fig. 8 (bottom)). In contrast to the other three methods outputting probabilities for each cluster label, kNN just outputs the predicted cluster label and we therefore did not compute the AUC for this method. We observe that the $\mathrm{LSTM}_{\mathrm{Seq}}$ and the $\mathrm{LSTM}_{\mathrm{End}}$ models clearly outperform the RF method at the beginning of the game (e.g., at level 3: $\mathrm{AUC}_{\mathrm{RF}} = 0.71$, $\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{Seq}}} = 0.75$, $\mathrm{AUC}_{\mathrm{LSTM}_{\mathrm{End}}} = 0.73$). Also in the middle of the game between levels 4 and 6, the RNN models show a higher AUC than the RF method (e.g., at level 5: $\mathrm{AUC}_{\mathrm{RF}} = 0.82$,

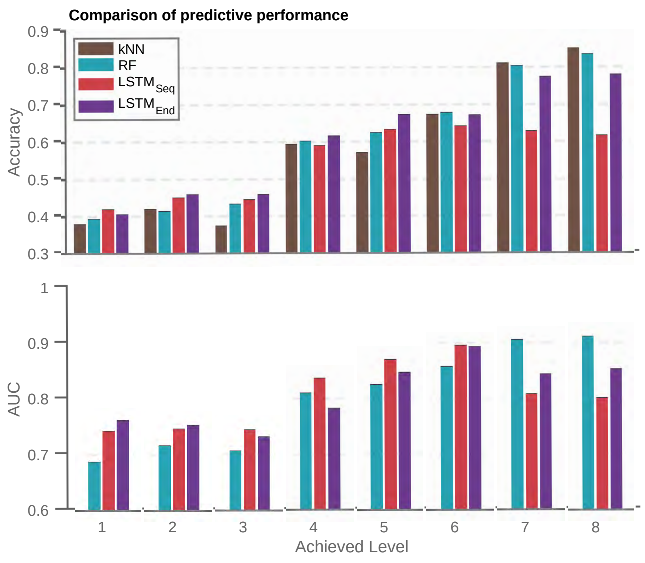

Figure 8: Accuracy (top) and AUC (bottom) of kNN, RF, $\mathrm{LSTM}_{\mathrm{Seq}}$, and $\mathrm{LSTM}_{\mathrm{End}}$ by achieved level. The RNNs achieve similar or better performance than the traditional classifiers up to level 6.

$\text{AUC}_{\text{LSTM}_{\text{Seq}}} = 0.89$, $\text{AUC}_{\text{LSTM}_{\text{End}}} = 0.87$) with the exception of level 4. When looking at the whole sequence, we observe a similar picture as for the accuracy: the AUC of the RF method is clearly higher than the AUC of the RNN models (at level 8: $\text{AUC}_{\text{RF}} = 0.91$, $\text{AUC}_{\text{LSTM}_{\text{Seq}}} = 0.80$, $\text{AUC}_{\text{LSTM}_{\text{End}}} = 0.85$).

As already mentioned before we assume that the worse performance of the RNN models at the end of the game (level 7 and level 8) is due to the fact that the input sequences for the RNN models become too long. Note that while most students manage to reach level 5 within a reasonable time frame, the lengths of the complete sequences vary significantly between the students. The lower accuracy and AUC of the RNN models at the end of the game are not an issue in our case because we are interested in accurate predictions early in the game. While the RNN models show only a slightly increased accuracy in comparison to the traditional methods at the beginning and in the middle of the game, they consistently achieve a higher AUC up to level 6, demonstrating their robustness towards class imbalance.

5. DISCUSSION

OELEs consitute a promising approach for learning. Ideally, the students learn the concepts and principles of a domain more deeply through exploration than if they are simply taught the principles and practice applying them. However, it has been shown that only a part of the students are able to effectively explore the space [20, 31, 24, 17]. Providing guidance to struggling students is therefore essential for educational success.

Because OELEs allow the user to freely interact with the content, traditional student modeling approaches cannot directly be applied to provide adaptation to the student. Adaptation based on detected student behavior and strategies is therefore a promising approach. Previous work has used offline clustering to detect different student types, followed by online classification of new students [15, 16, 11].

In this paper, we focused on the task of online classification, i.e., predicting the student type (or behavior) early on during interaction with the environment to provide targeted guidance as early as possible. In contrast to previous work applying standard classifiers such as k-nearest-neighbor [15, 16, 11], we suggested the use of RNN models for the online classification task. While previous research has investigated the use of RNN models to classify the students according to their problem-solving behavior based on their whole sequence of interactions [1], to classify changing learner behaviors over time [26], or to detect affective states over time [4], we explored the possibility of using RNN models to predict a student's cluster label (fixed over time) as early as possible. We have extensively evaluated a variety of RNN models and compared their predictive performance to the performance of k-nearest neighbor and random forest classifiers. We have used the different levels of the game as specific time points for evaluation as they pose realistic time points for interventions. We have trained RNN models to predict the cluster label at each time step as well as RNN models optimized for predicting the cluster label at each level of the game.

Not unexpectedly, the RNN models trained per level as well as the traditional classifiers outperform the models trained for predicting the whole sequence at the higher levels (level 7 and 8) of the game. This is due to the averaging effect of the performance of the 'sequence' RNN models: during training, the loss is computed for each time step of the sequence. Nevertheless, for level 4 and 5, which provide promising time points of intervention both in terms of accuracy of the different models as well as in terms of timing of intervention, the LSTM_{Seq} model with 16 hidden dimensions reaches similar performance as the other approaches. While this model does not outperform traditional approaches regarding prediction accuracy, it provides the potential for further adapting intervention. As the model is able to predict the cluster label at each time step, it is possible to provide the intervention at different points in time for different students depending on how sure the model is about the cluster label of the student. [21] have for example used a simple heuristic to at each time step decide whether the model should continue to see further time steps before outputting a final decision. While exhibiting the same accuracy, classification happened on average at an earlier point in time. This earlier classification allowed to provide targeted interventions sooner.

While the LSTM_{End} model with 16 hidden dimensions is also outperformed by the traditional classifiers at level 7 and 8 of the game, it shows a higher prediction accuracy than the kNN and RF classifiers for the first levels. Our results further demonstrate that the LSTM_{End} model with 16 hidden nodes outperforms the RF classifier regarding the AUC. This is especially important, because the AUC is not biased by imbalanced data sets.

We have also investigated different architectures for the RNN models. Our results demonstrate no large difference in the performance of LSTM and GRU models. However, due to their lower complexity, GRU models are more efficient and take less time to train the LSTM models. While this was not an issue for our small data set, it should be considered when training on larger data sets.

Due to the relatively small number of samples, we were able to only train shallow models with one or two hidden layers, not fully exploiting the advantages of deep learning. Furthermore, we also had to keep the number of dimensions of the hidden state low. However, the results achieved on our small data set are promising and we assume that the RNN models would perform even better on larger data sets.

In the future, we plan to design and test targeted interventions for the different clusters. Furthermore, we will collect a substantially larger data set to enable the training of deep neural networks using raw feature input only. Finally, we also plan to design and train the classifier such that scaffolded interventions can be delivered.

6. REFERENCES

[1] B. Akram, W. Min, E. N. Wiebe, B. W. Mott, K. Boyer, and J. C. Lester. Improving Stealth Assessment in Game-based Learning with LSTM-based Analytics. In *Proc. EDM*, pages 208–218, 2018.

[2] S. Amershi and C. Conati. Combining Unsupervised and Supervised Classification to Build User Models for Exploratory Learning Environments. *Journal of Educational Data Mining*, pages 18–71, 2009.

[3] G. Barata, S. Gama, J. Jorge, and D. Goncalves. Early Prediction of Student Profiles Based on Performance and Gaming Preferences. *IEEE Transactions on Learning Technologies*, 9(3):272–284, 2016.

[4] A. F. Botelho, R. S. Baker, and N. T. Heffernan. Improving sensor-free affect detection using deep learning. In *Proc. AIED*, pages 40–51, 2017.

[5] H. Cen, K. R. Koedinger, and B. Junker. Is Over Practice Necessary? -Improving Learning Efficiency with the Cognitive Tutor through Educational Data Mining. In *Proc. AIED*, pages 511–518, 2007.

[6] H. Cen, K. R. Koedinger, and B. Junker. Comparing Two IRT Models for Conjunctive Skills. In *Proc. ITS*, pages 796–798, 2008.

[7] F. Chollet et al. Keras. `https://keras.io`, 2015.

[8] A. T. Corbett and J. R. Anderson. Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *UMUAI*, 4(4):253–278, 1994.

[9] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-Term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691, 2017.

[10] Y. Fang, K. Shubeck, A. Lippert, Q. Cheng, G. Shi, S. Geng, J. Gatewood, S. Chen, C. Zhiqiang, P. Pavlik, J. Frijters, D. Greenberg, and A. Graesser. Clustering the Learning Patterns of Adults with Low Literacy Skills Interacting with an Intelligent Tutoring System. In *Proc. EDM*, pages 348–384, 2018.

[11] L. Fratamico, C. Conati, S. Kardan, and I. Roll. Applying a Framework for Student Modeling in Exploratory Learning Environments: Comparing Data Representation Granularity to Handle Environment Complexity. *International Journal of Artificial Intelligence in Education*, 27(2):320–352, 2017.

[12] C. Geigle and C. Zhai. Modeling MOOC Student Behavior With Two-Layer Hidden Markov Models. In *Proc. L@S*, pages 205–208, 2017.

[13] J. P. González-Brenes and J. Mostow. Topical Hidden Markov Models for Skill Discovery in Tutorial Data. *NIPS - Workshop on Personalizing Education With Machine Learning*, 2012.

[14] T. Hofmann and J. M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(1):1–14, 1997.

[15] S. Kardan and C. Conati. A Framework for capturing distinguishing user interaction behaviours in novel interfaces. In *Proc. EDM*, pages 159–168, 2011).

[16] T. Käser, A. G. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross. Cluster-Based Prediction of Mathematical Learning Patterns. In *Proc. AIED*, pages 389–399, 2013.

[17] T. Käser, N. R. Hallinen, and D. L. Schwartz. Modeling Exploration Strategies to Predict Student Performance Within a Learning Environment and Beyond. In *Proc. LAK*, pages 31–40, 2017.

[18] T. Käser, S. Klingler, A. G. Schwing, and M. Gross. Beyond Knowledge Tracing: Modeling Skill Topologies with Bayesian Networks. In *Proc. ITS*, pages 188–198, 2014.

[19] T. Käser and D. L. Schwartz. Detection and Analysis of Inquiry Strategies related to Transfer and Academic Achievement. *Journal of Artificial Intelligence in Education*, Under Review.

[20] J. S. Kinnebrew, K. M. Loretz, and G. Biswas. A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, 5(1), 2013.

[21] S. Klingler, T. Käser, A. G. Busetto, B. Solenthaler, J. Kohn, M. von Aster, and M. Gross. Stealth Assessment in ITS - A Study for Developmental Dyscalculia. In *Proceedings of the Int.'l Conference on Intelligent Tutoring Systems (ITS)*, pages 79–89, 2016.

[22] T. Lange, V. Roth, M. L. Braun, and J. M. Buhmann. Stability-based validation of clustering solutions. *Neural Comput.*, 16(6):1299–1323, 2004.

[23] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning Word Vectors for Sentiment Analysis. In *Proc. HLT*, pages 142–150, 2011.

[24] R. E. Mayer. Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *American Psychologist*, pages 14–19, 2004.

[25] S. Mojarad, A. Essa, S. Mojarad, and R. S. Baker. Data-Driven Learner Profiling Based on Clustering Student Behaviors: Learning Consistency, Pace and Effort. In *Proc. ITS*, pages 130–139, 2018.

[26] Z. A. Pardos, C. Hu, P. Meng, M. Neff, and D. Abrahamson. Classifying Learner Behavior from High Frequency Touchscreen Data Using Recurrent Neural Networks. In *Proc. UMAP*, pages 317–322, 2018.

[27] Z. A. Pardos, S. Tang, D. Davis, and C. V. Le. Enabling Real-Time Adaptivity in MOOCs with a Personalized Next-Step Recommendation Framework. In *Proc. Learning @ Scale*, pages 23–32, 2017.

[28] P. I. Pavlik, H. Cen, and K. R. Koedinger. Performance Factors Analysis - A New Alternative to Knowledge Tracing. In *Proc. AIED*, pages 531–538, 2009.

[29] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. *Proc. ICML,*, pages 727–734, 2000.

[30] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, and J. Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of NIPS*, pages 505–513, 2015.

[31] J. L. Sabourin, L. R. Shores, B. W. Mott, and J. C. Lester. Understanding and predicting student self-regulated learning strategies in game-based learning environments. *International Journal of Artificial Intelligence in Education*, 23(1):94–114, 2013.

[32] R. Sawyer, J. Rowe, R. Azevedo, and J. Lester. Filtered Time Series Analyses of Student Problem-Solving Behaviors in Game-based Learning. In *Proc. EDM*, pages 229–238, 2018.

[33] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, 2016.

[34] Y. Wang and J. Beck. Class vs. Student in a Bayesian Network Student Model. In *Proc. AIED*, pages 151–160, 2013.

[35] M. V. Yudelson, K. R. Koedinger, and G. J. Gordon. Individualized Bayesian Knowledge Tracing Models. In *Proc. AIED*, pages 171–180, 2013.