Inducing Stealth Assessors from Game Interaction Data

Wookhee Min[1(✉)], Megan H. Frankosky[1], Bradford W. Mott[1], Eric N. Wiebe[1], Kristy Elizabeth Boyer[2], and James C. Lester[1]

[1] Center for Educational Informatics, North Carolina State University, Raleigh, NC 27695, USA
{wmin, rmhardy, bwmott, wiebe, lester}@ncsu.edu
[2] Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL 32601, USA
keboyer@ufl.edu

Abstract. A key untapped feature of game-based learning environments is their capacity to generate a rich stream of fine-grained learning interaction data. The learning behaviors captured in these data provide a wealth of information on student learning, which stealth assessment can utilize to unobtrusively draw inferences about student knowledge to provide tailored problem-solving support. In this paper, we present a long short-term memory network (LSTM)-based stealth assessment framework that takes as input an observed sequence of raw game-based learning environment interaction data along with external pre-learning measures to infer students' post-competencies. The framework is evaluated using data collected from 191 middle school students interacting with a game-based learning environment for middle grade computational thinking. Results indicate that LSTM-based stealth assessors induced from student game-based learning interaction data outperform comparable models that required labor-intensive hand-engineering of input features. The findings suggest that the LSTM-based approach holds significant promise for evidence modeling in stealth assessment.

Keywords: Game-based learning environments · Stealth assessment · Deep learning · Computational thinking · Educational games

1 Introduction

Recent years have seen a growing interest in intelligent game-based learning environments because of their potential to effectively promote learning and engagement [1]. These environments simultaneously integrate the adaptive pedagogical functionalities of intelligent tutoring systems with the engaging interactions provided by digital games [2, 3]. Research has begun to explore student modeling for game-based learning environments including modeling student knowledge [4] and students' progression towards learning goals [5] following work on student-adaptive learning featuring tailored narratives, feedback, and problem-solving support [6].

E. André et al. (Eds.): AIED 2017, LNAI 10331, pp. 212–223, 2017.
DOI: 10.1007/978-3-319-61425-0_18

Stealth assessment [4] is a game-based assessment framework based on evidence-centered design (ECD) [7]. ECD features task, evidence and competency models for diagnostic measurement of multiple aspects of students' proficiency and performance. Built on the three models presented in ECD, stealth assessments utilize a rich stream of student interactions (i.e., an evidence model) with various problem-solving tasks (i.e., a task model) in game-based learning environments, to draw inferences about student knowledge and skills (i.e., a competency model). The evidence model provides the connections between the competency model and the stream of low-level observations, enabling the competency model to update the appropriate competencies related to the task being performed. In contrast to typical formative assessments, stealth assessment has the potential to not only create a valid, reliable evidence model utilizing observed sequences of detailed learning behaviors, but also to perform assessments of a wide range of constructs in an unobtrusive, invisible way, with the aim of providing useful feedback to students and teachers to enhance learning and inform instruction [4, 8].

A key challenge posed by stealth assessment is how to effectively handle both cyclical causalities between actions and events in the gameworld and temporal relationships characterized within learning behaviors. Students are likely to deliberately choose their next action by referring to the current task, their previous actions, and any feedback they received on their previous actions in the gameworld. Despite the popularity of utilizing evidence rules, which define a set of salient features that are indicative of specific student competencies in the evidence model, previous work based on evidence rules often ignores these complex relationships found within student learning behaviors [4, 9, 10]. Furthermore, these features are often hand-engineered, so they are domain expert-dependent, labor-intensive, and domain-specific.

As an alternative to manually devising an evidence model, an approach that automatically extracts patterns and learns predictive features from sequences of raw player actions would be more scalable, less labor-intensive, and would enable the induction of evidence models that directly represent student learning processes without sacrificing causal, temporal relationships. In this work, we investigate long short-term memory networks (LSTMs) [11], a type of gated recurrent neural network, for automating the creation of the evidence model without requiring hand-authored evidence rules and statistical models. LSTMs automatically extract salient features from temporal data and effectively preserve a longer-term memory by operating three gates featured in the network. Results of an evaluation suggest that LSTM-based stealth assessors directly induced from students' interactions with a game-based learning environment show significant promise for stealth assessment.

2 Related Work

Intelligent game-based learning environments are situated at the intersection of (1) digital games that increase students' motivation through rich settings (e.g., compelling plots, engaging characters) in virtual environments, and (2) intelligent tutoring systems that foster students' learning through tailored scaffolding and context-sensitive feedback. Recent work in game-based learning environments explores a broad

spectrum of subject matters ranging from high school mathematics [12], to middle school computer science [13], anti-bullying [14], language and culture learning [3], and science inquiry [15], among others.

Stealth assessment can play an important role in game-based learning environments. Previous work on stealth assessment based on evidence-centered design uses sequences of students' interactions with the learning environment to dynamically assess students' knowledge. For stealth assessment, various families of machine learning techniques have been investigated. Kim and colleagues [9] investigated Bayesian network-based evidence modeling, which requires two primary steps: (1) defining targeted competency and observable variables and building a directed graphical model, and (2) specifying the conditional probabilities between parent nodes and corresponding child nodes. Falakmasir et al. presented the SPRING data analysis pipeline that does not require costly domain knowledge engineering [16]. Specifically, SPRING trains two hidden Markov models (HMMs), one for high-performing and the other for low-performing students per game level. Two log-likelihoods of an observed sequence of student events are computed based on the two HMMs, and finally the difference between the two log-likelihoods for each game level is used as an independent variable for a linear regression model that predicts post-test scores. In our previous work, we presented *DeepStealth* [13], a framework based on deep neural networks [17] for stealth assessment. DeepStealth uses a deep feedforward neural network (FFNN)-based evidence modeling approach, in which the multi-level, hierarchical representations of the input data are learned through the training process of deep networks. While the last two approaches have an advantage over the Bayesian network-based approach by requiring less domain expert knowledge for evidence modeling, the competency model (e.g., competency model variables, dependencies between variables) is not designed at the same level of granularity as the Bayesian network and thus provide less fine-grained insight into concept mastery. While DeepStealth uses manually engineered features (e.g., features produced by expert-authored evidence rules), the LSTM-based approach introduced here fully automates the process of evidence modeling by directly utilizing raw game interaction data (i.e., a sequence of low-level actions).

3 ENGAGE Game-Based Learning Environment

ENGAGE (Fig. 1) is a game-based learning environment designed to introduce computational thinking to middle school students. It features a rich immersive 3D storyworld built with the Unity multi-platform game engine. The ENGAGE curriculum was developed by adapting the AP® Computer Science Principles course learning objectives [18] for U.S. middle school students (ages 11–13). A central aim of the curriculum and game-based learning environment is to promote computational thinking and problem-solving processes that involve abstraction and algorithmic thinking, and allow students to effectively use computational tools for data analysis, modeling, and simulations [19]. In addition to providing a foundation for advanced computer science work in high school, the problem-solving activities and computational challenges within the game are designed to increase middle school student's interest in computer science.

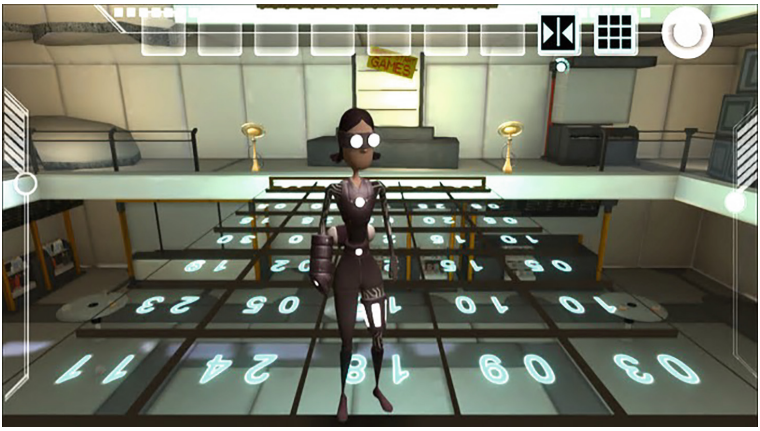

Fig. 1. ENGAGE game-based learning environment.

In the game, students play the protagonist who is sent to rescue an underwater research facility. As students progress through the game, they discover that all of the computing devices within the facility have been commandeered by a nefarious researcher. Students navigate through a series of interconnected rooms, each of which presents students with a set of computational challenges they must solve by either programming devices or operating devices in reference to the programs already written for the devices. Programmable devices are programmed using a visual programming language, in which visual blocks are linked together [13]. Finally, support is provided throughout the game by a cast of non-player characters who help progress the narrative and offer clues to assist students in solving the computational challenges.

One of the levels in the game, the Digital World, allows students to explore how binary sequences are used to represent digital data. The work presented in this paper focuses on students' problem-solving activities within this level. To complete a set of binary learning tasks, students must find the binary representation of the base-ten number stored in the binary lock device (Fig. 2, Left). Specifically, students must review an existing program (Fig. 2, Right) associated with the binary lock device, flip

Fig. 2. (Left) A binary lock device that students must unlock. The white tiles indicate the bits are 1, whereas black tiles denote 0. The current binary number is 01110 and the corresponding base-ten number, 14, is displayed on the device as immediate feedback. (Right) The visual programming interface displaying the binary lock's program.

binary tiles on the binary lock device to change the binary sequence (Fig. 2, Left), and execute its program. If the binary sequence matches the base-ten number stored in the program, the current binary lock device opens upon execution and the player can move on to a previously inaccessible area in the room. Through these tasks, students learn about the concept of bits in binary numbers and the weight assigned to each bit.

In this work, we analyze 191 students' interaction data (101 males, 88 females, 2 unreported) from a teacher-led deployment of ENGAGE in four public middle school classrooms. Prior to beginning the Digital World unit, and immediately following the unit, students completed online pre- and post-test assessments measuring computer science attitudes [20], self-efficacy [21], and content knowledge (e.g., binary representation). Students achieved improvements in content knowledge covered in the Digital World unit, and a paired *t*-test comparing pre-test (M = 0.43, SD = 0.21) to post-test (M = 0.59, SD = 0.24) indicated that students' learning gains were statistically significant with a sizable effect size, $t(185) = 12.25$, $p < .001$, $d = .70$, where 186 out of 191 students took both the pre- and post-knowledge tests. These external learning measures are used as predictive features for our evidence models, along with the game interaction data.

4 LSTM-Based Stealth Assessment Framework

For a stealth assessment framework to be scalable to a broad range of learning environments, it must be able to easily accommodate a wide range of domain-specific features. Focusing on this aspect, we first describe how our work is framed in evidence-centered design (ECD) [7] and then turn to our LSTM-based stealth assessment framework. From an ECD perspective, the three models are summarized as follows:

- *Task Model:* We use 11 binary-lock solving tasks from the Digital World unit, the objective of which is finding the binary representation that matches the base-ten number specified in the program.
- *Evidence Model:* Observed sequences of actions in the game reveal students' competencies. A *generic feature set* is used to represent actions. For ENGAGE, there are 19 possible actions, and thus 19 distinct features are used to represent each action using one-hot encoding. In addition to the game interaction evidence, students' five pre-test scores on the knowledge assessment, self-efficacy, and three measures of computer science attitudes are utilized as evidence. An LSTM-based evidence model informs the competency model in order to update students' competency levels.
- *Competency Model:* Following our previous work [13], we examine one competency model variable with respect to students' overall knowledge about binary representation, where the actual labels for their competency levels are acquired from students' post-test performance.

For domain independence, scalability, and robust performance, the evidence model supports a generic feature set as well as missing data. The low-level generic feature set in the evidence model can represent any types of action without being bound to a

specific domain, thereby yielding enhanced scalability for the stealth assessment framework. We use a *single* generic feature set to represent actions in this work, but the framework can support multiple feature sets depending on the design of actions in the learning environment (e.g., "*clicking the first binary tile*" can be represented using two distinct feature sets: the action-type feature set that contains *click*, and the action-argument feature set that contains *first binary tile*).

In this work, the binary learning tasks allow 19 possible actions, including 11 *pairing* actions[1] associated with 11 devices described in the task model (e.g., binary lock device in Fig. 2, Left), 5 *bit-click* actions (e.g., clicking a binary tile in Fig. 2, Left), two actions for operating the programming interface (*open* and *close* in Fig. 2, Right), and a *program execution* action to run the device's program.

The evidence model is designed to consider students whose data (either external pre-test scores or task activities) is partially missing. For example, it is possible that a student missed a class and has only partial gameplay data or did not complete some pre-tests prior to playing the game. To formulate the external learning measure evidence from missing pre-test data, we perform mean imputation using a mean score of other students' scores for the specific pre-test. On the other hand, in cases where students did not solve a specific task in the game, the game evidence is generated by linking any observed learning activities, skipping the unsolved tasks. For example, if a student completed only two tasks (T_1 and T_3) and missed one task (T_2) in-between, the activities for T_1 and T_3 are linked to generate a data instance, ignoring T_2. Since it is not uncommon for a student to be absent from class within a multi-week intervention, this specific design for the evidence model is necessary to broaden tailored learning support to all students who participated in the learning activities.

For the competency model, students' competencies are represented by their post-test performance on the knowledge assessment items for binary representations. The competencies are defined based on a tertile split ('high', 'medium', or 'low') with respect to post-test scores on the assessment, and thus this stealth assessment task is cast as a three-class classification problem that predicts one's competency level using an LSTM-based stealth assessor.

4.1 Long Short-Term Memory Networks

Long short-term memory networks (LSTMs) (Fig. 3A) are a variant of recurrent neural networks (RNNs) that are specifically designed for sequence labeling of temporal data. Traditional RNNs have faced significant challenges with respect to vanishing or exploding gradients during training deep networks unfolded in time [22]. The three gating units (input gate, output gate, and forget gate) featured in LSTMs enable modeling long-term dependencies within temporal sequences by allowing gradient information to flow over many time steps. LSTMs have achieved state-of-the-art performance in a diverse set of computational sequence-labeling tasks, including speech recognition and machine translation [23].

[1] Within the game, students must pair their virtual in-game computer with devices before they can manipulate or view a device's programs.

In an implementation of LSTMs, the input gate (i_t), forget gate (f_t), candidate value of the memory cell ($\tilde{c}_t$), and output gate (o_t) at time t are computed by Eqs. 1–4, respectively, in which W and U are weight matrices for transforming the input (x_t) at time t and the cell output (h_{t-1}) at time $t-1$, b is the bias vector of each unit, and σ and *tanh* are the logistic sigmoid and hyperbolic tangent function, respectively:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{4}$$

As described in Eq. 5, the current memory cell's state (c_t) is calculated by modulating the current memory candidate value ($\tilde{c}_t$) via the input gate (i_t) and the previous memory cell state (c_{t-1}) via the forget gate (f_t). Through this process, a memory cell decides whether to keep or forget the previous memory state and regulates the candidate of the current memory state via the input gate. Once again, the current memory cell state (c_t) is controlled by the output gate (o_t) to compute the cell activation (h_t) of the LSTM block at time t. This step is described in Eq. 6:

$$c_t = i_t \tilde{c}_t + f_t c_{t-1} \tag{5}$$

$$h_t = o_t \, tanh(c_t) \tag{6}$$

Lastly, we use the final memory cell output vector (h_t) to predict the class label for stealth assessment, which is the competency level of the student. This step is executed in a softmax layer (top-right in Fig. 3A), which is interpreted as a calculation of posterior probabilities of the possible class labels. The LSTM is end-to-end trainable, where all the parameters such as W, U, and b are machine-learned using backpropagation through time.

4.2 Configuring LSTMs for Evidence Modeling

The LSTM's input, x_t, represents the evidence that a student reveals at time t. As noted above, the evidence model considers students' pre-learning measures in addition to actions in the game. These two types of variables feature different dynamics: actions are sequential and discrete, whereas the external learning measures are static and numeric, since they are measured prior to starting the game. Figure 3B describes how we encode these two different types of variables into a trainable input (x_t) at time t. First, we concatenate the integer index of the action at time t (a_t) with the five static external learning measures (e_1–e_5) to generate the original input ($input_t$). While scores for external learning measures (e.g., e_1) can be directly utilized by the LSTMs because their relative, numeric values are meaningful, the action index, a_t, should be reformulated since its discrete value does not represent a magnitude.

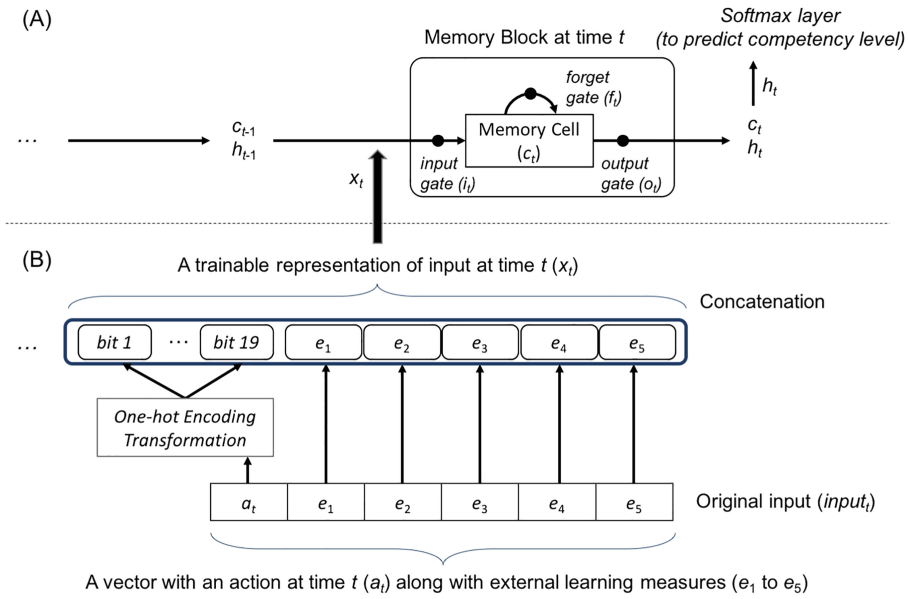

Fig. 3. (A) An illustration of an LSTM memory block that features three gating units and a memory cell [22]. (B) An illustration of how an original input ($input_t$) is transformed to a trainable format (x_t). The discrete action variable, a_t, is one-hot encoded into a 19-dimensional vector using bit 1 to 19, and then the induced vector is concatenated with numeric external learning measure variables (e_1 to e_5) to create the final input, x_t.

To address this issue, we use one-hot encoding to represent actions. One-hot encoding creates a bit vector whose length is the number of the actions, where only the associated action bit is on (i.e., 1), while all other bits are off (i.e., 0). Since we consider 19 distinct actions in ENGAGE, an action (e.g., a_t) is represented with a 19-dimensional vector. The final input (x_t) is generated by concatenating the one-hot encoded action representations with the five external learning measures, and thus the input is a 24-dimensional vector. Like actions in the input, the output of LSTMs should also be represented using one-hot encoding, due to its discrete nature. Since the number of possible competency levels is three in our work, the output is represented using a three-dimensional one-hot vector.

Given this encoding of actions, the next step is to devise an encoding for action sequences. Suppose that a student performed three actions and achieved the competency level, 'high'. We generate x_1, x_2, and x_3 based on our input encoding approach. A naïve method to generate a sequence is creating one from the list of actions, [x_1, x_2, x_3], along with the target label 'high'. Another approach to generate sequences is using *sequence subsampling*. The sequence subsampling method can generate more sequences for the same case. For the same example, a subsampling method can produce three sequences, [x_1], [x_1, x_2], and [x_1, x_2, x_3], all with the same target label of 'high', by accumulating actions sequentially. While the naïve approach creates only one training example (i.e., one sequence), this subsampling approach can create as many training

examples as the number of actions per student (three sequences in this example). Since actions in a sequence represent a student's dynamic learning progress to achieve the final competency, we adopt the subsampling method that induces fine-grained training examples.

Finally, as with many other machine learning techniques, an effective configuration of network hyperparameters for LSTMs often must be empirically determined. There are several categories of hyperparameters to consider, including optimization (e.g., optimizer, learning rate), model structure (e.g., the number of hidden units, initialized weights), and training criterion (e.g., regularization terms, loss function) [24]. In this work, we adopt a grid-search on a model structure-based hyperparameter, the number of hidden units, which has the most significant influence on predictive performances of LSTMs among others in student goal recognition work [5]. We explore five values for the hyperparameter: 80, 100, 120, 140 and 160. Other than this, we investigate a single-layer LSTM with a softmax layer for classifying given sequences of actions, adopt a mini-batch gradient descent with the mini-batch size of 128, set the dropout rate [25], a regularization parameter, to 0.75, and utilize categorical cross entropy for the loss function and the Adam stochastic optimizer [26]. Finally, the training process stops early if the validation score has not improved within the last seven epochs. In this work, 10% of the training data is used to determine early stopping, while 90% is utilized for supervised training, leaving the test set purely unseen. The maximum number of epochs is set to 100. For devising LSTM-based evidence models, we use Keras [27], a python-based modular neural networks library.

5 Evaluation

We evaluate evidence models' predictive accuracy with 10-fold student-level cross-validation. The same data split is used for a fair comparison with the competitive baseline approaches. In this empirical evaluation, 191 students' gameplay data along with their external pre-learning measures are investigated, where 35,571 data instances are generated for training LSTM-based evidence models, following the sequence subsampling technique. We compare the LSTM model to the previous state-of-the-art deep feedforward neural network pre-trained with stacked denoising autoencoders (FFNN) [13], support vector machine (SVM), and naïve Bayes model (NB). As discussed, unlike our LSTM models, the three competitive baseline models utilize four salient game features engineered by domain experts, including the number of binary tile flips, the number of binary tile double flips (a binary tile flipped and then immediately flipped again), the number of times the device programs are executed, and the amount of time students spent in the programming interface [13]. Also, for these three baseline models, in case that the gameplay data is partially missing, mean imputation is performed per game feature as done for missing pre-learning measures, since these models take fixed-size inputs. All four evidence modeling approaches utilize the same set of external learning measures as additional evidence.

For each computational approach, the best model configurations are identified in the process of 10-fold cross-validation. Similar to the grid search method applied for the LSTMs, we grid-search a set of hyperparameters for FFNNs, SVMs and NBs.

For FFNNs, we explore two hyperparameters, the number of hidden layers (from one to five) and corruption rate (four randomly chosen values), while freezing some other hyperparameters (e.g., 40 hidden units per layer, softmax for the output activation function). We examine two hyperparameters that are popularly explored for optimization: the penalty parameter (C) and gamma (γ) for SVMs with a radial basis function [28]. C is chosen from {1, 10, 50, 100}, and γ is chosen from {0.0005, 0.001, 0.005, 0.01, 0.05}. Finally, for NBs, we investigate two distributions (normal distribution and kernel smoothing density estimate) to fit models for the data.

Table 1. Average accuracy rates of the LSTMs, FFNNs, SVMs, and NBs. {columns : rows} for the four machine learning techniques indicate {number of hidden units}, {number of hidden layers : corruption rate}, {gamma : penalty parameter}, and {distribution}, respectively. The highest accuracy rate is marked in bold for each technique.

LSTMs	**80**	**100**	**120**	**140**	**160**
	58.1%	56.1%	58.6%	**63.9%**	60.7%
FFNNs	**1**	**2**	**3**	**4**	**5**
0.20	61.9%	59.1%	56.6%	57.6%	59.7%
0.39	56.6%	61.3%	60.7%	56.5%	55.5%
0.69	59.1%	54.0%	**62.9%**	52.3%	54.5%
0.82	58.1%	59.7%	57.1%	55.0%	49.8%
SVMs	**0.0005**	**0.001**	**0.005**	**0.01**	**0.05**
1	50.8%	55.5%	59.1%	58.1%	56.0%
10	58.6%	59.2%	58.6%	59.2%	58.6%
50	59.2%	**59.7%**	56.6%	57.1%	58.1%
100	59.2%	56.6%	58.7%	58.7%	58.1%
NBs	**Normal**	**Kernel**			
	48.1%	41.6%			

Table 1 reports the average accuracy rates across different hyperparameter configurations for each machine learning technique from cross-validations. Overall, the highest performing LSTMs (the number of hidden units: 140) achieve 63.9% accuracy rate, which outperforms the highest performing models from FFNNs (62.9%), SVMs (59.7%) and NBs (48.1%) as well as the majority class baseline (41.9%).

In addition to the predictive performance improvement, the LSTM-based stealth assessment has two notable benefits over the baseline approaches. First, the capacity to handle various lengths of action sequences, effectively learning sequential patterns, and performing sequence labeling per action points towards LSTMs as being a viable solution for stealth assessment. For instance, as opposed to the FFNN-based approach that takes as input a fixed size of input features generated using the entire sequence of actions, LSTMs can sequentially make a prediction per action, and thus enable dynamic, run-time formative assessments on student competencies. Second, LSTMs directly utilize raw game interaction data dispensing with the need for manually engineering features to induce stealth assessors. This characteristic constitutes considerable benefits over the other models, since the feature engineering process is not only labor and time-intensive, but

also impedes scalability of the stealth assessment framework to other learning environments due to the domain-specificity of the engineering process. It is noteworthy that the LSTMs directly utilizing low-level inputs achieve the highest accuracy without leveraging expert knowledge.

6 Conclusions and Future Work

This paper has introduced a novel LSTM-based stealth assessment framework that shows promise for accurately assessing learners' competency levels. Using data collected from multi-week classroom deployments of a game-based learning environment for middle grade computational thinking, we conducted an evaluation of four stealth assessment induction approaches that predict student post-competencies. The results suggest that LSTM-based stealth assessors outperform the previous state-of-the-art approach, deep feedforward neural networks pre-trained with stacked denoising autoencoders, as well as support vector machines and naïve Bayes models, with respect to predictive accuracy of students' post-competencies. This result is notable in that the LSTM-based evidence models were induced directly using raw game interaction data, whereas the other models were devised using domain-expert engineered features. Together with the sequence modeling capability, the LSTM-based stealth assessment framework offers the potential to serve as the foundation for formative assessment that operates dynamically, unobtrusively, and is readily applicable to various learning environments. In the future, it will be important to investigate stealth assessor model optimizations and regularizations for further improving performance and informing decision making for adaptive scaffolding. It will be also important to measure the stealth assessors' early prediction performance to evaluate their capacity for formative assessment. It will also be important to design a granular set of competencies for stealth assessors to be more diagnostic and provide fine-grained pedagogical support to further enhance student learning.

Acknowledgments. This research was supported by the National Science Foundation under Grant CNS-1138497 and Grant DRL-1640141. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Lester, J., Ha, E., Lee, S., Mott, B., Rowe, J., Sabourin, J.: Serious games get smart: intelligent game-based learning environments. AI Mag. **34**(4), 31–45 (2013)
2. Jackson, T., McNamara, D.: Motivation and performance in a game-based intelligent tutoring system. J. Educ. Psychol. **105**(4), 1036–1049 (2013)
3. Johnson, L.: Serious use of a serious game for language learning. Int. J. Artif. Intell. Educ. **20**(2), 175–195 (2010)
4. Shute, V.J., Ventura, M.: Measuring and supporting learning in games: stealth assessment. In: Computer Games, Simulations & Education. The MIT Press, Cambridge (2013)
5. Min, W., Mott, B., Rowe, J., Liu, B., Lester, J.: Player goal recognition in open-world digital games with long short-term memory networks. In: International Joint Conference on Artificial Intelligence, pp. 2590–2596 (2016)

6. Brusilovsky, P., Millán, E.: User models for adaptive hypermedia and adaptive educational systems. In: The Adaptive Web, pp. 3–53 (2007)
7. Mislevy, R., Steinberg, L., Almond, R.: On the structure of educational assessment. Meas. Interdiscip. Res. Perspect. **1**(1), 3–62 (2003)
8. Rosenheck, L., Lin, C.Y., Klopfer, E., Cheng, M.T.: Analyzing gameplay data to inform feedback loops in The Radix Endeavor. Comput. Educ. **111**, 60–73 (2017)
9. Kim, Y.J., Almond, R.G., Shute, V.J.: Applying evidence-centered design for the development of game-based assessments in physics playground. Int. J. Test. **16**(2), 142–163 (2016)
10. Smith, A., Aksit, O., Min, W., Wiebe, E., Mott, B.W., Lester, J.C.: Integrating real-time drawing and writing diagnostic models: an evidence-centered design framework for multimodal science assessment. In: Micarelli, A., Stamper, J., Panourgia, K. (eds.) ITS 2016. LNCS, vol. 9684, pp. 165–175. Springer, Cham (2016). doi:10.1007/978-3-319-39583-8_16
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1–32 (1997)
12. Kebritchi, M., Hirumi, A., Bai, H.: The effects of modern mathematics computer games on mathematics achievement and class motivation. Comput. Educ. **55**(2), 427–443 (2010)
13. Min, W., Frankosky, M.H., Mott, B.W., Rowe, J.P., Wiebe, E., Boyer, K.E., Lester, J.C.: DeepStealth: leveraging deep learning models for stealth assessment in game-based learning environments. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M.F. (eds.) AIED 2015. LNCS, vol. 9112, pp. 277–286. Springer, Cham (2015). doi:10.1007/978-3-319-19773-9_28
14. Vannini, N., Enz, S., Sapouna, M., Wolke, D., Watson, S., Woods, S., Dautenhahn, K., Hall, L., Paiva, A., André, E., Aylett, R.: "FearNot!": a computer-based anti-bullying-programme designed to foster peer intervention. Eur. J. Psychol. Educ. **26**(1), 21–44 (2011)
15. Nelson, B.C., Kim, Y., Foshee, C., Slack, K.: Visual signaling in virtual world-based assessments: the SAVE Science project. Inf. Sci. **264**, 32–40 (2014)
16. Falakmasir, M.H., Gonzalez-Brenes, J.P., Gordon, G.J., DiCerbo, K.E.: A data-driven approach for inferring student proficiency from game activity logs. In: ACM Conference on Learning at Scale, pp. 341–349 (2016)
17. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
18. AP® Computer Science Principles Draft Curriculum Framework. http://www.csprinciples.org/. Accessed 05 Feb 2017
19. K–12 Computer Science Framework. http://www.k12cs.org/. Accessed 05 Feb 2017
20. Wiebe, E., Williams, L., Yang, K., Miller, C.: Computer science attitude survey. Comput. Sci. **14**(25), 1–86 (2003)
21. Chen, G., Gully, S.M., Eden, D.: Validation of a new general self-efficacy scale. Organ. Res. Methods **4**(1), 62–83 (2001)
22. Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks. Studies in Computational Intelligence, vol. 385. Springer, Heidelberg (2012)
23. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2014)
24. Bengio, Y.: Practical recommendations for gradient-based training of deep architectures. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 437–478. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35289-8_26
25. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
26. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
27. Chollet, F.: Keras. https://github.com/fchollet/keras. GitHub Repository. Accessed 05 Feb 2017
28. Keerthi, S.S., Lin, C.-J.: Asymptotic behaviors of support vector machines with Gaussian kernel. Neural Comput. **15**(7), 1667–1689 (2003)