IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 3, 2019, accepted January 26, 2020, date of publication February 6, 2020, date of current version February 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2972167

Individualized AI Tutor Based on Developmental Learning Networks

WOO-HYUN KIM AND JONG-HWAN KIM (Fellow, IEEE)
School of Electrical Engineering, KAIST, Daejeon 34141, South Korea

Corresponding author: Jong-Hwan Kim (johkim@rit.kaist.ac.kr)

This work was supported in part by the Institute for Information and Communications Technology Promotion (IITP) funded by the Ministry of Science and ICT (MSIT), Korea Government (Research on Adaptive Machine Learning Technology Development for Intelligent Autonomous Digital Companion) under Grant 2016-0-00563.

ABSTRACT In recent years, in the field of education technology, artificial intelligence tutors have come to be expected to provide individualized educational services to help learners achieve high levels of academic success. To this end, AI tutors need to be able to understand the current status and preferences of a learner and then suggest appropriate learning contents accordingly. However, it is challenging to monitor learner status and preferences continually and to recommend appropriate educational services. In this paper, we propose an individualized AI tutor as an integrated system of three developmental learning networks (DLNs) by extending a deep adaptive resonance theory (Deep ART) network, a neural network capable of incremental learning. Specifically, the learner status DLN is able to easily add new input channels about learner status without disrupting existing classifiers. The learner preference DLN is to categorize learner preferences based on frequency as well as sequence of events. The learner experience DLN is updated to immediately reflect alteration of the educational effectiveness in the current classification. Our AI tutor is currently embedded in a commercialized mobile application for teaching the Korean language to children. Experimental results show that the AI tutor application efficiently helps children learn the Korean language.

INDEX TERMS Adaptive resonance theory, artificial intelligence tutor, individualized education, machine learning, online mobile application.

I. INTRODUCTION

Educational technology (Edutech) gives learners endless opportunities to learn new things and allows learners to customize their learning, taking into account their abilities and mobility. The Edutech industry is growing at 24% annually, and the global markets will reach over $252 billion by 2020 [1] thanks to the state-of-the-art technologies [2]. In recent years, artificial intelligence tutors are receiving attention as virtual teachers offering a personalized approach for each student through efficient data analysis. For example, AI tutors can track and visualize learning outcomes to support self-learning and can replace expensive private teachers in certain subjects, such as language learning [3]–[5].

Identifying the learner's present levels and status, and periodically reviewing education plans is the most important to maximize educational effectiveness [6]. Therefore, AI tutors providing individualized educational services have to track the learner educational levels and status and recommend educational goals and contents [7]. Besides this, AI tutors need to adapt to learning progress and review learning plans; as such, AI tutors should be able to learn through their own experiences, incrementally.

The ART network, a neural network, is appropriate for implementing an individualized AI tutor since it was originally developed to learn data incrementally [8]. Initially, the ART network was only able to deal with binary input patterns. By the work of many researchers, the ART network has been improved so that it can handle complicated input patterns [9]–[11]. Recently, the ART network was utilized for classification and recommendation [12], [13], hybrid data regression [14], topological clustering [15], implementation of long term memory [16]–[19], and interactive learning [20]. However, ART networks still have limitations in direct application to AI tutors. First, ART networks cannot add new input channels. It is difficult to predefine all input channels. In order to implement AI tutors, therefore, ART networks should be able to take into account new types of information

The associate editor coordinating the review of this manuscript and approving it for publication was Xian Sun.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

about the learner status observed during the learning process. Second, ART networks should be extensible to handle the frequency of events of input channels. Although conventional ART networks can deal with a temporal sequence of events, learner preferences are related not only to the sequence but also to the frequency of events. Third, the classifications of ART networks needs to be flexible both in expanding and contracting ways. Real-time alteration of the educational effectiveness should be immediately reflected in the current classification of the ART network.

In this paper, we propose an AI tutor that provides individualized education programs to learners considering their current status and preferences. To this end, we develop an integrated system of three developmental learning networks (DLNs): learner status DLN, learner preference DLN, and learner experience DLN based on the Deep ART network [16]. Specifically, we propose a new algorithm for each DLN; individual algorithms respectively deal with the limitations as mentioned above. First, an input channel addition algorithm is developed to adapt dynamically changing input channels in the learner status DLN. For the learner preference DLN, an event bundle encoding algorithm is proposed to consider the frequency of events. Finally, an alternative template learning algorithm is applied to reflect the educational effectiveness of the learner experience DLN in real-time.

By conducting a set of extensive simulations, we show that our proposed algorithms efficiently resolve the limitations of existing ART networks. Our individualized AI tutor implemented using the proposed DLNs is currently embedded on a commercialized mobile application for teaching the Korean language to children. In this application, the AI tutor helps children to learn the Korean language in the most efficient and interesting way. The application is available at https://play.google.com/store/apps/details?id=com.h2kresearch.preciousHangul&hl=en_US

The remainder of this paper is organized as follows. Section II introduces the preliminaries for better understanding. In Section III, we present the overall structure of the proposed individualized AI tutor, along with a detailed description of it. In Section IV, we analyze simulations and conduct a set of comprehensive experiments, using mobile applications to verify the performance of the proposed system. Finally, conclusion and further work follow in Section V.

II. PRELIMINARIES

A. FUSION ART NETWORK

Fusion ART network is basically an unsupervised incremental learning model, which has multiple input channels [10]. The basic structure of the Fusion ART network is shown in Fig. 1. The learning processes are described in the following.

Complement Coding: Let ${}^k\mathbf{I} = ({}^kI_1, {}^kI_2, \ldots, {}^kI_n)$ be the input vector of the k-th channel ${}^kF_1$ of the input field where $k \in \{1, 2, \ldots, c\}$ is the channel number, c is the number of channels, and ${}^kI_i \in [0, 1]$. Then, each input channel generates

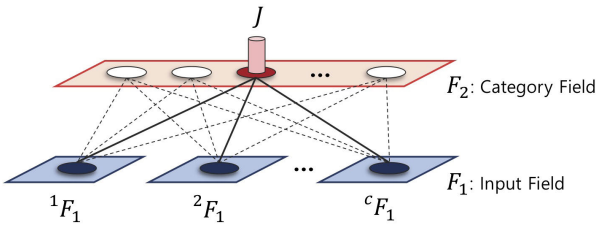

FIGURE 1. Basic structure of fusion ART.

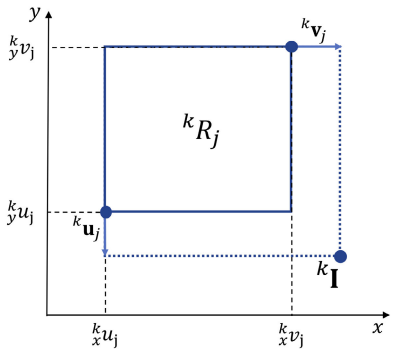

FIGURE 2. Rectangular area that a weight vector represents.

an activity vector ${}^k\mathbf{x} = ({}^k\mathbf{I}, {}^k\bar{\mathbf{I}})$ that concatenates the input vector ${}^k\mathbf{I}$ and its complement vector ${}^k\bar{\mathbf{I}} = \mathbf{1} - {}^k\mathbf{I}$. The purpose of this process is to normalize the input vector and to prevent weights from converging to zero.

Code Activation: The j-th node in the category field F_2 is activated from $\mathbf{x} = [{}^1\mathbf{x}, {}^2\mathbf{x}, \ldots, {}^c\mathbf{x}]$, by the following choice function:

$$T_j = \sum_{k=1}^{c} {}^k\gamma \frac{|{}^k\mathbf{x} \wedge {}^k\mathbf{w}_j|}{\alpha + |{}^k\mathbf{w}_j|} \tag{1}$$

where T_j is an activation value of the j-th node, ${}^k\mathbf{w}_j$ is a weight vector associated with the j-th category node, and the k-th input channel, ${}^k\gamma \in [0, 1]$, is a contribution parameter that indicates how much each input channel contributes to the choice function, $\alpha = 0^+$ is a choice parameter, $\wedge$ is the fuzzy AND operator defined as $(p_i \wedge q_i) \equiv \min(p_i, q_i)$, and $|\cdot|$ is the norm defined as $|\mathbf{p}| \equiv \sum_i p_i$.

The weight vector ${}^k\mathbf{w}_j = ({}^k\mathbf{u}_j, {}^k\bar{\mathbf{v}}_j)$ represents a rectangular area ${}^kR_j$ bounded by ${}^k\mathbf{u}_j$ and ${}^k\mathbf{v}_j$, which are a bottom-left corner point (closest to the origin) and a top-right corner point (farthest from the origin), respectively, as shown in Fig. 2. $|{}^k\mathbf{x} \wedge {}^k\mathbf{w}_j| = |{}^k\mathbf{w}_j|$ if the input ${}^k\mathbf{I}$ is located in ${}^kR_j$, or $|{}^k\mathbf{x} \wedge {}^k\mathbf{w}_j| < |{}^k\mathbf{w}_j|$ if the input ${}^k\mathbf{I}$ is outside of ${}^kR_j$.

Code Competition: The category node that has the largest choice function value is selected and indexed as

$$J = \underset{j \in S_{F_2}}{\text{argmax}} \{T_j\} \tag{2}$$

where $S_{F_2} = \{1, 2, \ldots, m\}$ is the set of indexes of nodes in the category field F_2.

Template Matching: In the template matching process, the most appropriate category for the activity vector is found

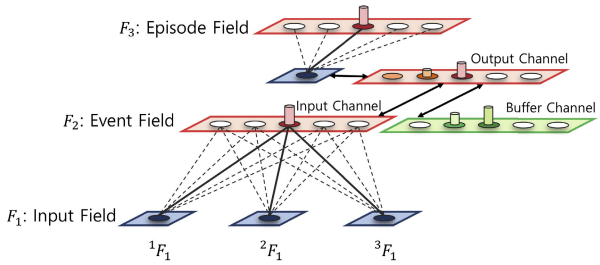

FIGURE 3. Basic structure of Deep ART network.

to update the weight of the category. After the index J is chosen, system checks if the match function ${}^k m_J$ meets the following resonance condition:

$$ {}^k m_J = \frac{|{}^k\mathbf{x} \wedge {}^k\mathbf{w}_J|}{|{}^k\mathbf{x}|} \geq {}^k\rho \tag{3} $$

where ${}^k\rho$ is a vigilance parameter that determines the resonance.

For all k, if the resonance condition is satisfied, which means that ${}^k\mathbf{x}$ belongs to the J-th category, the weight vector, ${}^k\mathbf{w}_J$, is updated. If there is a certain k that does not satisfy the resonance condition, i.e., ${}^k m_j < {}^k\rho$, the current J node is rejected, T_j is set to zero, and the same processes are repeated from the code competition process. If no node satisfies the resonance condition, a new category node is created, and an associated weight vector is initialized by ${}^k\mathbf{x}$.

Template Learning: If the resonance condition is satisfied, the weight vector ${}^k\mathbf{w}_j$ is updated as follows:

$$ {}^k\mathbf{w}_J^{(new)} = (1 - {}^k\beta){}^k\mathbf{w}_J^{(old)} + {}^k\beta({}^k\mathbf{x} \wedge {}^k\mathbf{w}_J^{(old)}) \tag{4} $$

where ${}^k\beta \in [0, 1]$ is the learning rate of the k-th channel. By this operation, the rectangular area ${}^kR_j$ represented by ${}^k\mathbf{w}_J^{(old)}$ expands up to an area that includes both the previous rectangular area and the input ${}^k\mathbf{I}$.

B. DEEP ART NETWORK

Deep ART network was developed to learn temporal information for biologically inspired episodic memory [16]. This network has one more layer than the Fusion ART network, as depicted in Fig. 3. The bottom network is the same as the Fusion ART network with a similar learning procedure, but the category field F_2 is re-described as an event field. The event field represents a temporal sequence of events with the input, buffer and output channels. The top network categorizes the sequence as an episode, where the episodic field takes the input from the event field using only a single channel.

Episode Encoding: The Deep ART network subdivides the event field into three channels. Let ${}^i\mathbf{y}$, ${}^b\mathbf{y}$, and ${}^o\mathbf{y}$ be vectors of the input channel, the buffer channel, and the output channel in the event field, respectively. The input channel vector ${}^i\mathbf{y}$ represents which event node is activated by setting the element indicating the activated node to 1 and other elements

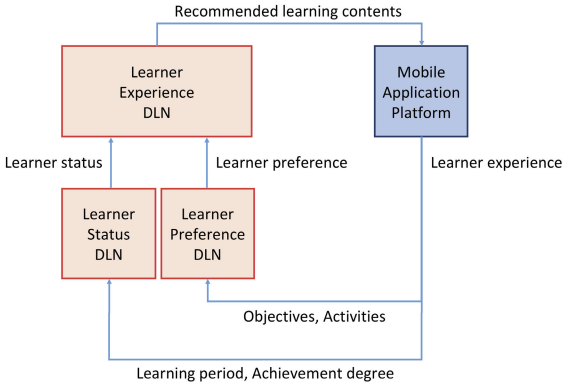

FIGURE 4. Overall structure of individualized AI tutor.

to 0 as follows.

$$ {}^i y_j = \begin{cases} 1, & \text{if } j = J. \\ 0, & \text{otherwise.} \end{cases} \tag{5} $$

The buffer channel vector ${}^b\mathbf{y}$ is assigned to buffer the output channel vector ${}^o\mathbf{y}$, i.e. ${}^b\mathbf{y} = {}^o\omega \cdot {}^o\mathbf{y}^{(old)}$. ${}^o\omega$ is the weight of the output channel. The output channel vector is calculated by combining the input channel vector and the buffer channel vector, as follows:

$$ {}^o\mathbf{y}^{(new)} = {}^i\omega \cdot {}^i\mathbf{y} + {}^b\omega \cdot {}^b\mathbf{y} \tag{6} $$

where ${}^i\omega$ and ${}^b\omega$ are the weights of the input channel and buffer channel, respectively. As a duplicate event is not overwritten due to the additional channels, temporal sequences of events can be stored and retrieved without the loss of information [16].

Episode Learning: The output vector from the event field, which is encoded as an episode, is stored in the top network. As there is only a single input channel, the choice function T_p for the p-th node in the episode field F_3 is as follows:

$$ T_p = \frac{|\mathbf{y} \wedge \mathbf{w}_p|}{|\mathbf{w}_p|} \tag{7} $$

where $\mathbf{w}_p$ is the weight vector for the p-th node in the episode field.

III. INDIVIDUALIZED AI TUTOR

A. OVERALL STRUCTURE

The proposed individualized AI tutor is based on the integration of three DLNs, as shown in Fig. 4. By integrating three different types of networks, the proposed system can identify the learner status and provide individualized educational services simultaneously. Each DLN has its own role. The two DLNs below categorize the learner status and preferences, and the above DLN recommends appropriate learning contents. The roles and features of each DLN are described sequentially in the following subsections.

B. LEARNER STATUS DLN

The learner status DLN is designed to categorize the current academic status of a specific learner. Learning status

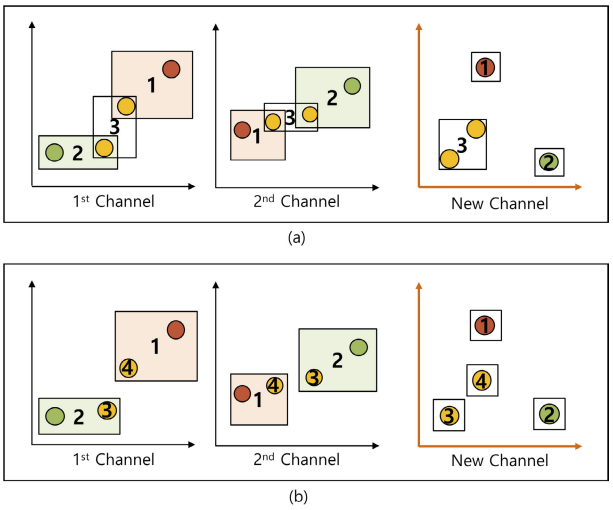

FIGURE 5. Input channel addition algorithm: Red, green, yellow circles represent the new inputs that come into the network after a new channel has been added. (a) Conventional algorithm. (b) Input channel addition algorithm.

is an academic achievement of learning objectives during specific learning periods. The proposed network is based on the Fusion ART network, which stores associative mappings of input patterns across multiple channels. Each input channel in the learner status DLN represents a particular pattern space that distinguishes the learner status. However, the conventional networks are designed to handle only fixed input channels, without taking into account changes in input channels [10]. In practical applications, it is often possible to observe new metrics or features. It is also hard to define all channels of input features before the learning process. For these reasons, we propose an input channel addition algorithm to adapt to dynamic environments and continue developmental learning.

Input channel addition algorithm: Fig. 5 shows the difference between the input channel addition algorithm and the existing one. When a new channel is added, the conventional learning process raises a problem that merges even inputs of different categories into the same category. The addition of a new channel means that a new pattern space has appeared, which becomes involved in associative mapping. Right after a new channel is added, there is no information about associative mapping between existing categories and the new channel. Therefore, it is necessary to use input data to find out the information about associative mapping. In Fig. 5, two categories, which are represented by red and green rectangles, have been created when only two channels exist. After a new channel is added, four inputs enter one by one. Each input consists of the three channels' vectors. In the proposed algorithm, yellow inputs are separated into two categories because those inputs belong to different categories in the prior two channels.

Once a new channel is added, the input channel addition algorithm deals with the existing channels and the new channel separately. A few nodes in the category field have already been active before input channels are changed. The input from

Algorithm 1 Input Channel Addition Algorithm

Input: $\mathbf{I} = [{}^1\mathbf{I}, {}^2\mathbf{I}, ..., {}^c\mathbf{I}, {}^a\mathbf{I}]$

1: Separate the existing and new channels from the input. $\mathbb{E} = \{{}^1\mathbf{I}, {}^2\mathbf{I}, ..., {}^c\mathbf{I}\}, \mathbb{N} = \{{}^a\mathbf{I}\}$
2: For the existing channels, calculate the complement code and the activation values T_j by using (1).
3: For the existing channels, find the nodes that satisfy the resonance condition as in (3), in order from the highest activation value to the lowest value.
4: **if** There is no node that satisfies the resonance condition. **then**
5: Create the $(m+1)$-th node, and initialize the weight ${}^k\mathbf{w}_{m+1} = ({}^k\mathbf{I}, {}^k\bar{\mathbf{I}}), k \in \{1, 2, \ldots, c, a\}$
6: **else**
7: Among the nodes that satisfy the resonance condition in Step 3, find the nodes that satisfy the resonance condition by the new channel.
8: **if** There is no node that satisfies the condition. **then**
9: Create the $(m+1)$-th node, and initialize the weight ${}^a\mathbf{w}_{m+1} = {}^a\mathbf{x}, {}^k\mathbf{w}_{m+1} = {}^k\mathbf{w}_J, k \in \{1, 2, \ldots, c\}$ J is the index of the most highly-activated node in Step 3.
10: **else**
11: Update the weight $\mathbf{w}_J$ for all $k \in \{1, 2, \ldots, c, a\}$ by using (4). J is the index of the node that satisfies the resonance condition for all channels in Step 7.
12: **end if**
13: **end if**

the existing channels is used to check whether it belongs to the activated nodes or not. If it doesn't belong to any activated node, the new node is created, which has the initial weight set to the input from both channels. If it belongs to the activated node, it is checked whether the input from the new channel also satisfies the resonance condition of the activated node. If the input from the new channel satisfies the resonance condition, the weight of the activated node is updated by the input. If there is no node satisfying the resonance condition, the new node is created, which has the initial weight of the new channel set to the input from the new channel. On the other hand, the initial weights of the existing channels are set to the weights of the node activated by the input from the existing channels. Therefore, by separating channels, the proposed algorithm maintains existing information and simultaneously performs associative mapping to new pattern space.

C. LEARNER PREFERENCE DLN

The learner preference DLN is developed to categorize the learning preference of a specific learner. Learning preference is a learning style that appears in the learning process and includes complicated characteristics such as learning habits, learning methods, and learning tips. The developed network is based on the Deep ART network, which deals with the

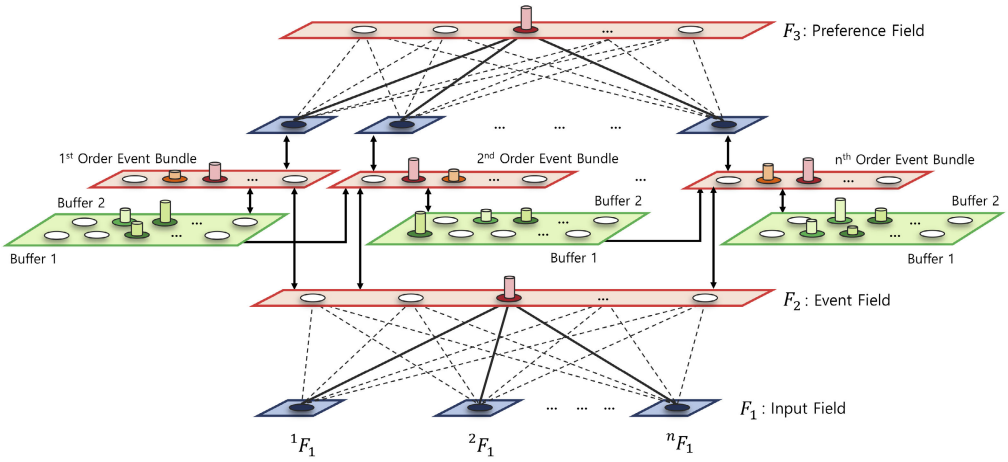

FIGURE 6. Structure of learner preference DLN.

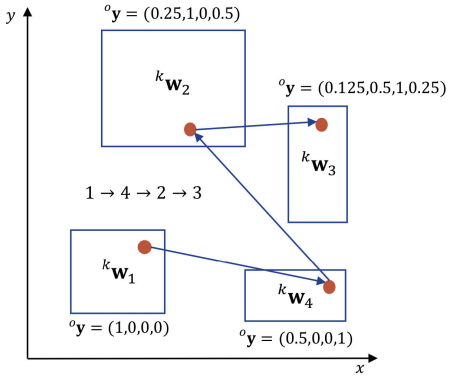

FIGURE 7. Example of the event sequence encoding: There are four events represented by the rectangle areas in the graph. As four events occur sequentially, the output vector $^o\mathbf{y}$ is calculated by using (6). $^i\omega$, $^b\omega$, $^o\omega$ are set to 1, 0.5, and 1, respectively.

temporal sequence of events. An event means a situation that the learner has met or a certain activity that the learner has done. High incidence of an event indicates that the learner prefers to learn the same way. However, conventional networks are not designed to deal with an accumulation of events. We devise the event bundle encoding algorithm to store the number of times events occur in an episode. Fig. 6 shows the structure of the learner preference DLN. The algorithm is delineated in the following.

Event bundle encoding algorithm: Fig. 7 shows an example of event sequence encoding in the Deep ART network [16]. The events occur according to the following sequence: 1, 4, 2, and 3. Although the event1 occurs only once at all of the sequences, the element values associated with event1 are all different. The event sequence encoding process cannot describe the number of times the events occur. To solve this issue, we propose the n-th order event bundle by using additional buffer layers.

The event bundles are defined as small segments included in the entire event sequence. One event bundle represents a partial pattern of consecutive events categorized from the bottom layer in the learner preference DLN. The 1st order event bundle is the same as each event that constitutes the entire sequence. The n-th order event bundle is a partial sequence that consists of n events. For example, if the sequence consists of 10 events, the 1st order event bundles are 10, but the 5th order event bundles are just 6.

Let $^i\mathbf{y} = (^iy_1, ^iy_2, \ldots, ^iy_m)$ denote the input channel vector. The 1st order event bundle $^e_1\mathbf{y}$ is the same as the input channel vector $^i\mathbf{y}$. When the J-th event occurs, elements of the input channel vector are as follows:

$$^iy_j = \begin{cases} 1, & \text{if } j = J. \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Similar to the Deep ART Network [16], the proposed algorithm uses buffer channels to encode the n-th order event bundles because the inputs are entered one by one. The difference is that the incidence is encoded, not the order of events, as follows:

$$^o\mathbf{y}^{(new)} = ^e_n\mathbf{y} + ^o\mathbf{y}^{(old)} \tag{9}$$

$$^o\bar{\mathbf{y}}^{(new)} = \frac{1}{N} \cdot ^o\mathbf{y}^{(new)} \tag{10}$$

where N is the number of n-th order event bundles included in the entire sequence. The normalized output channel vector $^o\bar{\mathbf{y}}$ is calculated by dividing the vector $^o\mathbf{y}$ by N.

The 1st order event bundle can be expandable to a higher-order event bundle. The 2nd order event bundle $^e_2\mathbf{y}$ has the same number of dimensions as the 1st order event bundle, but the number of events used for the calculation is different. Two consecutive events are used to calculate the 2nd order event bundle, as follows:

$$^e_2\mathbf{y} = ^i\omega \cdot ^i\mathbf{y} + ^b\omega \cdot ^b\mathbf{y} \tag{11}$$

$$^b\mathbf{y} = ^e_1\mathbf{y}^{(old)} = ^i\mathbf{y}^{(old)} \tag{12}$$

where $^i\omega, ^b\omega \in [0, 1]$, and $^i\omega + ^b\omega = 1$. Each 2nd order event bundle $^e_2\mathbf{y}$ represents the sequence of two events. As in (10), the final output channel vector $^o\mathbf{y}$ encodes the incidence of the

2nd order event bundles. As the event bundle expands from 1st order to 2nd order, two buffer channels are needed: one is for the 2nd order event bundle, and the other is for the output channel vector.

Identical to the expanded 2nd order event bundle, it is possible to expand the n-th order event bundle. The n-th order event bundle requires one buffer channel more than the $(n-1)$-th order event bundle, as follows:

$$ {}^{e}_{n}\mathbf{y} = {}^{i}\omega \cdot {}^{i}\mathbf{y} + {}^{b}\omega \cdot {}^{b}\mathbf{y} \quad (13) $$

$$ {}^{b}\mathbf{y} = {}^{e}_{n-1}\mathbf{y}^{(old)} \quad (14) $$

where ${}^{i}\omega$, ${}^{b}\omega \in [0, 1]$, and ${}^{i}\omega + {}^{b}\omega = 1$. There are n output channels in the event field to allow encoding of all n-th order event bundles simultaneously. For each output channel, one buffer channel is required to deal with sequential inputs. On the other hand, every output channel shares the input channel. The n-th order event bundle is calculated through the current input and the $(n-1)$-th order event bundle. Therefore, it is necessary to connect each event bundle to one higher-order event bundle by using an additional buffer channel.

D. LEARNER EXPERIENCE DLN

The learner experience DLN is designed to store the learning experience of a specific learner. Learning experiences are learning records that have been performed by learners whose status and preferences are identified. And they are also the learning outcomes that show the effect of recommended contents. New learning experiences perpetuate existing memories, but also sometimes edit memories about previous experiences. In education, for example, the education effect may be different, even with the same content for the same learner. Depending on the educational effectiveness, the network has to decide whether to recommend the learning content or not. To this end, the classifications of the network need to be flexible both in expanding and contracting ways. For the learner experience DLN, alternative template learning algorithm is devised.

Alternative template learning algorithm: The alternative template learning algorithm is different from the conventional template learning algorithm in that there are two weights between the one channel and one category. The learning experience updates these two weights simultaneously. Depending on the educational effectiveness, one expands and the other contracts the template. Expansion means memorizing the new data, while the contraction means discarding the existing data.

The learner experience DLN is based on the Fusion ART network. The network has three types of input channels. Two input channels deal with the clustering results of the learner status DLN and learner preference DLN. The last channel handles the learning content. In the learning content channel, ${}^{k}\mathbf{w}_{j}$ is a base weight vector associated with the j-th category node and the k-th input channel. Each base weights has two alternative templates: ${}^{k}\mathbf{w}_{jH}$ and ${}^{k}\mathbf{w}_{jL}$ are high and low effectiveness templates, respectively.

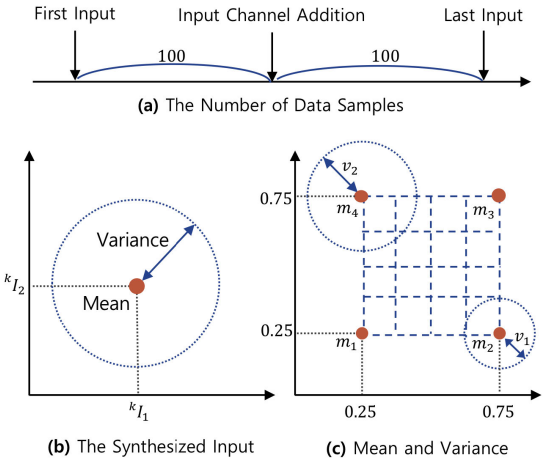

FIGURE 8. Simulation setup for the input channel addition algorithm: (a) Number of data samples. (b) Synthesized input. (c) Types of mean and variance.

When a high-effectiveness content ${}^{k}\mathbf{x}_{H}$ comes in, if the content belongs to the J-th category the alternative template learning algorithm is as follows:

$$ {}^{k}\mathbf{w}_{JH}^{(new)} = (1-\beta_I)\,{}^{k}\mathbf{w}_{JH}^{(old)} + \beta_I({}^{k}\mathbf{x}_{H} \wedge {}^{k}\mathbf{w}_{JH}^{(old)}) \quad (15) $$

$$ {}^{k}\mathbf{w}_{JL}^{(new)} = (1+\beta_O)\,{}^{k}\mathbf{w}_{JL}^{(old)} - \beta_O({}^{k}\mathbf{x}_{H} \wedge {}^{k}\mathbf{w}_{JL}^{(old)}) \quad (16) $$

where β_I and β_O are the learning rates for the internal division and outer division, respectively. According to (15), the high effectiveness template ${}^{k}\mathbf{w}_{JH}$ expands to the high-effectiveness content ${}^{k}\mathbf{x}_{H}$. On the other hand, the area of the low effectiveness template ${}^{k}\mathbf{w}_{JL}$ is reduced by (16).

When a low-effectiveness content ${}^{k}\mathbf{x}_{L}$ comes in, if the content belongs to the J-th category the alternative template learning algorithm is as follows:

$$ {}^{k}\mathbf{w}_{JH}^{(new)} = (1+\beta_O)\,{}^{k}\mathbf{w}_{JH}^{(old)} - \beta_O({}^{k}\mathbf{x}_{L} \wedge {}^{k}\mathbf{w}_{JH}^{(old)}) \quad (17) $$

$$ {}^{k}\mathbf{w}_{JL}^{(new)} = (1-\beta_I)\,{}^{k}\mathbf{w}_{JL}^{(old)} + \beta_I({}^{k}\mathbf{x}_{L} \wedge {}^{k}\mathbf{w}_{JL}^{(old)}) \quad (18) $$

where β_I and β_O are the learning rates for the internal division and outer division, respectively. The area of the high effectiveness template ${}^{k}\mathbf{w}_{JH}$ is reduced by (17). On the other hand, according to (18), the low effectiveness template ${}^{k}\mathbf{w}_{JL}$ expands to the low-effectiveness content ${}^{k}\mathbf{x}_{L}$.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. INPUT CHANNEL ADDITION ALGORITHM

The simulation analysis was conducted to verify the proposed input channel addition algorithm. Fig. 8 shows the experimental setup of the synthesized inputs and groups. The inputs were generated in 50 groups according to a combination of mean and variance values. Each group was composed of 200 input data. The inputs entered separately in two phases because an input channel was added to the network in the middle of the process.

Fig. 9 shows the index of the activated category corresponding to the input. After a new channel was added, the number of activated categories were increased rapidly.

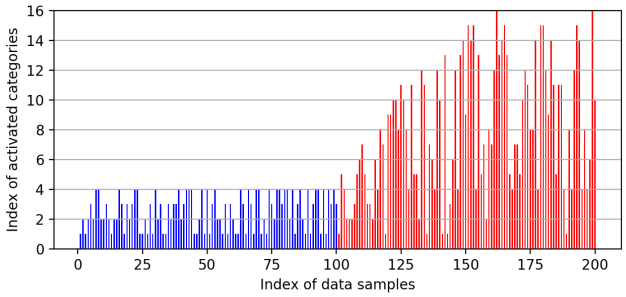

FIGURE 9. Index of activated categories.

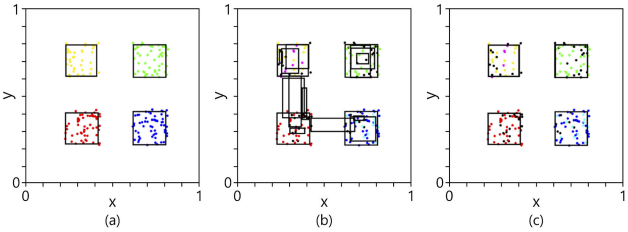

FIGURE 10. Clustering results: (a) Conventional algorithm without an additional channel. (b) Conventional algorithm with an additional channel. (c) Proposed algorithm with an additional channel.

The graph was blue until the 100th data sample. After that, it became red. There were only four categories before the input channel addition.

Fig. 10 shows that the proposed algorithm maintained the prior information. All graphs shows the categories generated according to the same inputs. Fig. 10(a) is the result without the input channel addition. Fig. 10(b) and (c) include the results after adding the new channel. In Fig. 10(a), there are only four categories; however, in other figures, there are more categories. This means that the different types of inputs entered are due to the new channel. However, inputs separated at first merged into the same new category when the proposed algorithm was not used, as shown in Fig. 10(b). Besides, some templates were shrunken compared with the original templates. On the other hand, the proposed algorithm still distinguished the inputs that had been categorized separately before. It seems to divide the original category into several sub-categories.

Fig. 11 shows the accuracy of the proposed algorithm. The accuracy was computed by comparing the category activated after input channel addition with that assigned before. Fig. 11(a) and (b) show the results when the variances were 0.05 and 0.1, respectively. The higher the variance is, the harder it is to cluster data. Moreover, the data had a higher density due to the position of means as x increased. In the graphs, the blue and red indicate the conventional method and the proposed algorithm, respectively. Most red had higher accuracy than blue. Although it was harder to cluster data in Fig. 11(b), red still had better performance.

B. EVENT BUNDLE ENCODING ALGORITHM

As shown in Fig. 12, the following inputs were synthesized to compare with the time sequence encoding. The synthesized

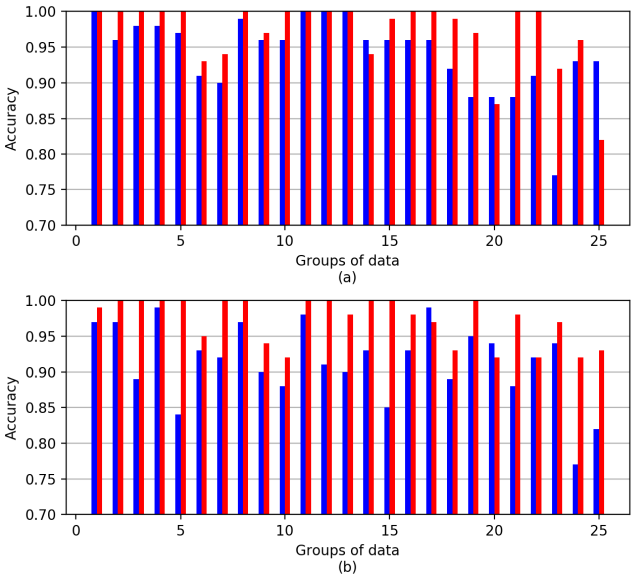

FIGURE 11. Accuracy results: Proposed algorithm and the conventional algorithm are red and blue, respectively. (a) $\upsilon = 0.05$. (b) $\upsilon = 0.1$.

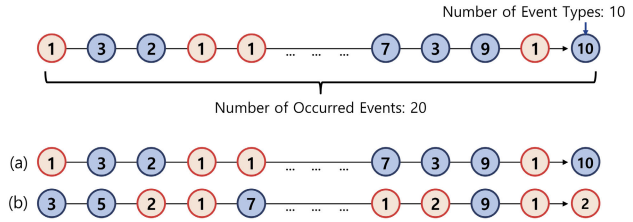

FIGURE 12. Synthesized event sequence: (a) Example of a sequence for encoding the 1st order event bundle. (b) Example of a sequence for encoding the 2nd order event bundle.

event sequence was composed of 20 events regarded as having virtually occurred. Each event might be one of the ten types of events. We defined ten experimental conditions for simulation analysis. The first five conditions were designed for the 1st order event bundle; others were for the 2nd order. Of the 20 events, the determination of the including rate of the target event was different in each condition.

We conducted the 1st order event bundle encoding and the time sequence encoding according to the first five conditions. In each condition, 1, 2, 4, 8, or 16 event1s were included. According to these conditions, 100 event sequences were randomly generated to verify that the event bundles stored the frequency of a single event. Fig. 13 shows a comparison result between the event bundle encoding and the time sequence encoding. In the figure, the red point represents the values of the first element of the event bundle encoding; the blue is the value of the first element of the time sequence encoding. The first element of each encoding is related to the event1. In the case of the event bundle encoding, the value increased in proportion to the number of event1s included in each sequence. However, the time sequence encoding did not represent the number of event1s included at all. As a result, only the event bundle encoding algorithm stored the frequency of a single event.

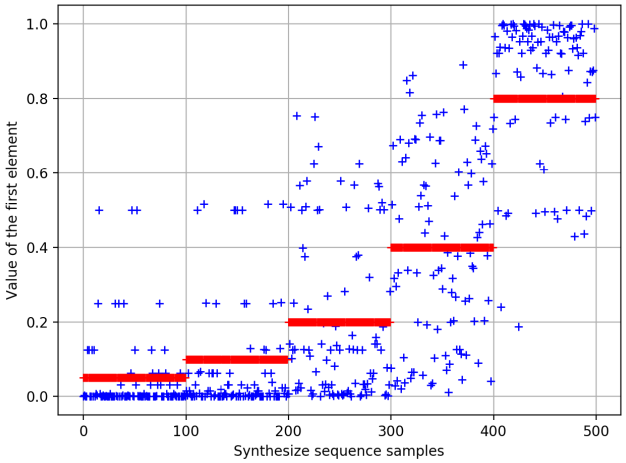

FIGURE 13. Comparison result between the 1st order event bundle encoding and the time sequence encoding: From the first to the 100th sequence sample, samples were generated according to the first condition. The samples included only one event1. From the 101st to 200th, just two event1s were included in the entire sequence. Every following 100 samples, samples were generated according to the different conditions, in the serial order mentioned below.

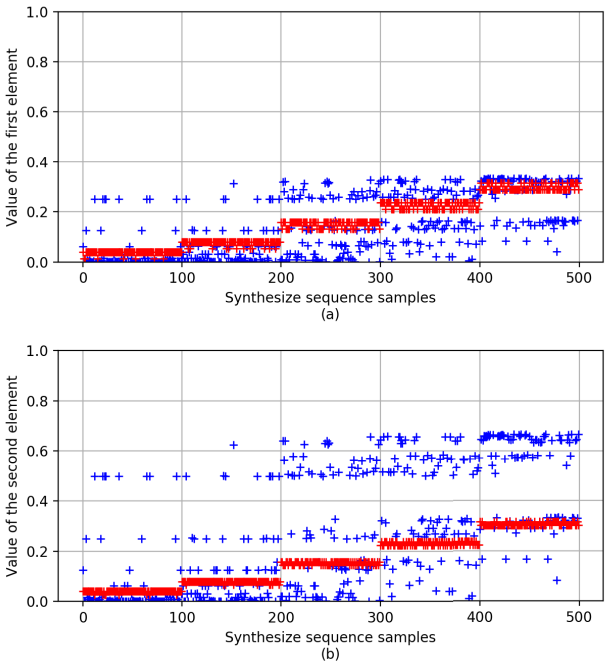

FIGURE 14. Comparison result between the 2nd order event bundle encoding and the time sequence encoding: Red points are associated with the 2nd order event bundle encoding. Blue points are associated with the time sequence encoding. (a) Value of the first element of the encoding vector. (b) Value of the second element of the encoding vector.

We also conducted the 2nd order event bundle encoding according to the last five conditions. In each condition, 1, 2, 4, 6, or 8 sequences composed of event1 and event2 were included, respectively. For each encoding result, Fig. 14(a) shows the value of the first element, and Fig. 14(b) shows the value of the second element. Similar to the results of the 1st order event bundle encoding, the values increased in proportion to the number of sequences composed of event1

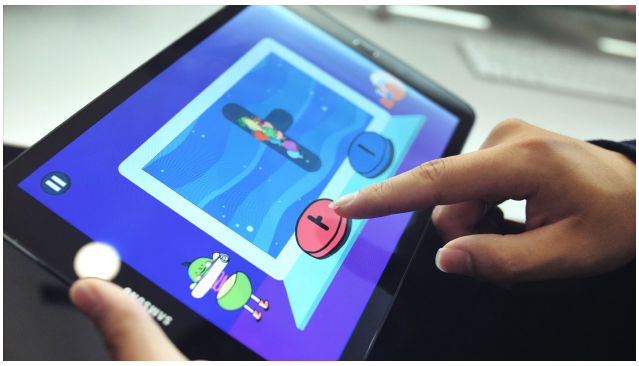

FIGURE 15. Mobile application teaching the Korean language for children.

FIGURE 16. Snapshots of educational games: Eight activities were used for the experiments. (a) Hearing. (b) Seeking. (c) Doing a puzzle. (d) Blowing up a balloon. (e) Playing with dinosaur. (f) Playing with slingshot. (g) Baking a cake. (h) Doing the laundry.

and event2. However, the time sequence encoding did not represent the number of the sequences at all. Therefore, the figure shows that the event bundle encoding algorithm can also store the frequency of a consecutive event.

C. MOBILE APPLICATION PLATFORM

Overview: We developed a mobile application to teach the Korean language to children (as shown in Fig. 15). The application has been offered in the Apple App Store and Google Play Store since October 2018 and currently consists of more than 30 educational games, as shown in Fig. 16. It was featured in the Apple App Store in March 2019 and has

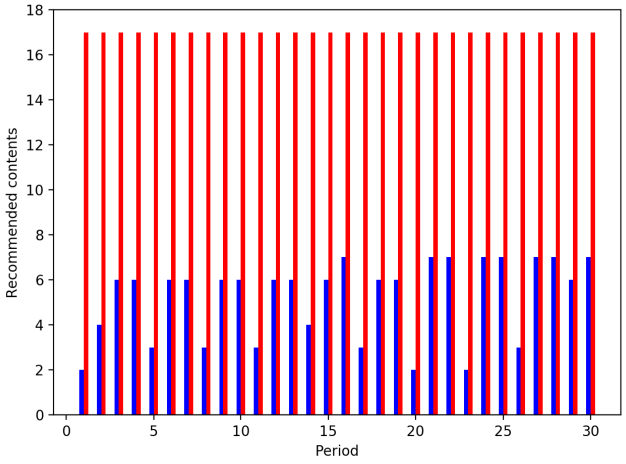

FIGURE 17. Comparison of the number of contents suggested between curriculum and AI tutor: Red bar represents the number of contents that the proposed AI tutor recommended. Blue bar represents the number of contents that are included in the curriculum at that period.

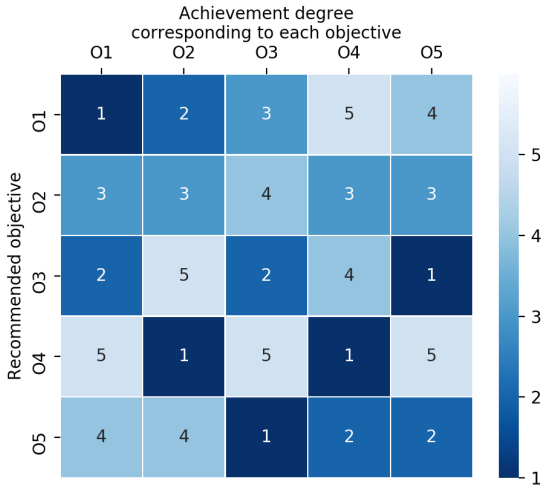

FIGURE 18. A matrix to check the learning objectives recommendation performance of the proposed AI tutor.

been top-ranked in the iPad Kids application category since then. The number of total users is over 30,000 and, 500 users are playing it daily. The total number of user learning data collected is more than 5,000,000.

In this experiment, only 800,000 learning data obtained from 1,000 users with high utilization were used to train our individualized AI tutor. The reason was that, to develop individualized AI tutors, it was appropriate to leverage data only from users with at least 30 days of usage. Learning database consists of five columns, e.g., User ID, Period, Objectives, Activities, and Achievement Degree, each of which corresponds to a user identifier, a learning date, a learning objective (five learning objectives categories in total), an educational game to teach the learning objective (eight educational game categories in total), and accumulated average achievement score vectors of learning objectives. For the learner status DLN, we used 800,000 training data of both the Period and Achievement Degree. For the learner preference DLN, 800,000 training data with the Activity were used. Here, to provide the activity frequency information to the network, the Activities input was a vector that consists of a maximum of 100 previous activities. For the learner experience DLN, however, only 230,000 training data with negative achievement were used to design our AI tutor, which recommends contents based on the learner weaknesses.

To determine how diversely the AI tutor recommended contents and how appropriate those contents were, we compared the performance of the AI tutor with a standard educational curriculum. The curriculum was developed by an expert with 15 years of experience in the Korean language education field and then reviewed by three experts with more than 10 years of experience in the field.

Comparison between Curriculum and AI Tutor: First, for each period, we compared the number of contents recommended by our AI tutor to the number of contents of the curriculum. As presented in Fig. 17, the recommended contents by our AI tutor included all the contents of the curriculum at each period. These results show that the AI tutor is capable of recommending the same contents as the curriculum suggests to an individual learner. In addition, the AI tutor can also suggest the other contents that the curriculum does not cover. This implies that the proposed AI tutor can consider more various types of learner status and preferences in comparison to the curriculum.

A learning content consists of a learning objective and a learning activity. There are five types of learning objectives in the Korean language education: vowel, consonant, double consonant, double vowel and final consonant. We checked whether the learning objectives of contents that the proposed AI tutor recommended were appropriate. Fig. 18 shows how learning objectives are recommended according to the achievement degree. In the figure, each column represents recommendation rankings from 1 to 5, where objective with ranking 1 is the most likely to be recommended to a learner whose weakness corresponds to the column's representative objective. For example, column O1 lists 1, 3, 2, 5, and 4 from the first row. Therefore, learners with weakness O1 were often recommended to learn the objective O1 of the first row whose recommendation rank is 1. Overall, we can see that our AI tutor takes into account the weaknesses of learners, and then recommends an appropriate objective to address that weakness.

In this experiment, we tested whether the proposed AI tutor appropriately reflects a learner's preferences when recommending activities. There are eight types of learning activities in all learning contents: such as hearing, seeking, doing a puzzle, blowing up a balloon, playing with dinosaur, playing with slingshot, baking a cake, doing the laundry. In Fig. 19, each column represents recommendation rankings from 1 to 8, where activity with ranking 1 is the most likely to be recommended to a learner whose preference corresponds to the column's representative preference. For example, column A2 has values of 5, 1, 2, 7, 3, 4, 8, and 6 listed in the first row.

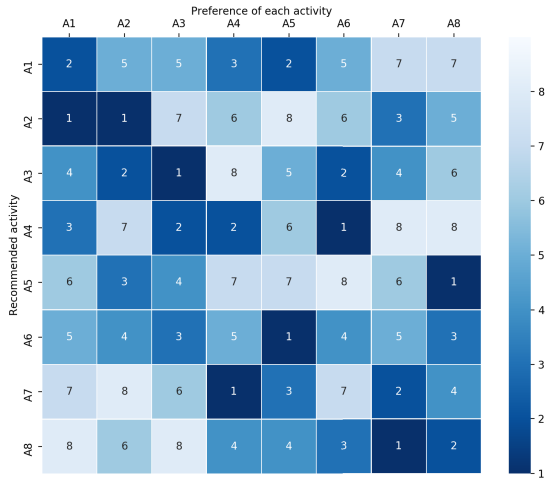

FIGURE 19. Matrix to check the learning activities recommendation performance of the proposed AI tutor.

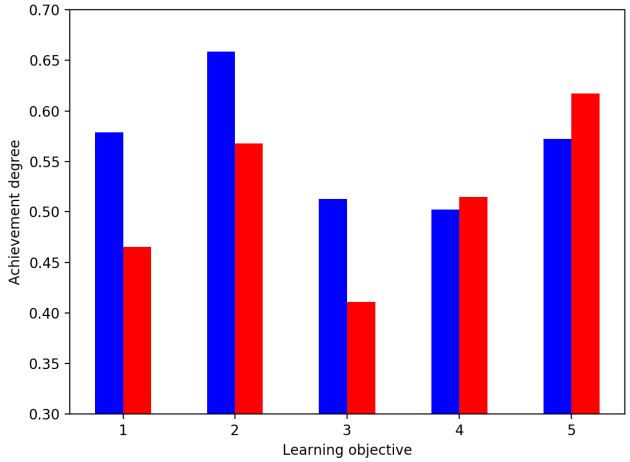

FIGURE 20. Alteration over time of the recommended learning objectives: Blue bar represents the achievement degree of each objective in the early stages of learning. On the other hand, red bar represents the achievement degree of each objective in the later stages of learning.

Therefore, learners with preference A2 are recommended to follow activity A2 of the second row, whose recommendation rank is 1. Overall, we can see that our AI tutor takes into account the preferences of the learner, and then recommends an appropriate activity reflecting such those preferences.

Fig. 20 shows the change over time of the recommended learning objectives. The higher the achievement degree was, the more frequently the objective is recommended. As can be seen in the blue bar, the AI tutor suggested primary learning contents (i.e., Objectives 1, 2, and 3) more in the early stages of learning. On the other hand, it suggested difficult learning contents (i.e., Objectives 4 and 5) more in the later stages of learning, as shown in the red bar. This result shows that the AI tutor recommended appropriate learning goals based on learner achievement.

V. CONCLUSION AND FURTHER WORK

This paper proposed a novel individualized AI tutor to help a learner achieve a high level of academic success. To consider the current learner status and preferences, we developed the AI tutor as a system integrating three DLNs by extending the Deep ART network. The proposed AI tutor was trained with 800,000 training sets collected from a commercialized mobile application teaching the Korean language. Our experimental results show that the proposed AI tutor can suggest to learner suitable learning contents that correspond to what the standard education curriculum presents. Besides this, we showed that our AI tutor can recommend appropriate learning contents even for learners to whom the standard curriculum cannot make suggestions. As further work, we will collect the individual learning data from the interaction between the proposed AI tutor and a learner through the mobile application platform to measure the effectiveness of a long-term education.

REFERENCES

[1] M. Watch. *Global Report Predicts Edtech Spend to Reach $252 BN by 2020*. Accessed: 2016. [Online]. Available: https://www.marketwatch.com/press-release/global-report-predicts-edtech%-spend-to-reach-252bn-by-2020-2016-05-25-4203228

[2] N. Ibrahim, H. Hammed, A. Zaidan, B. Zaidan, O. Albahri, M. Alsalem, R. Mohammed, A. N. Jasim, A. H. Shareef, and N. Jalood, "Multi-criteria evaluation and benchmarking for young learners' English language mobile applications in terms of LSRW skills," *IEEE Access*, vol. 7, pp. 146620–146651, 2019.

[3] C.-K. Hsu, G.-J. Hwang, and C.-K. Chang, "A personalized recommendation-based mobile learning approach to improving the reading performance of EFL students," *Comput. Edu.*, vol. 63, pp. 327–336, Apr. 2013.

[4] V. Bradac and B. Walek, "A comprehensive adaptive system for e-learning of foreign languages," *Expert Syst. Appl.*, vol. 90, pp. 414–426, Dec. 2017.

[5] Y. Lee, Y. Choi, J. Cho, A. R. Fabbri, H. Loh, C. Hwang, Y. Lee, S.-W. Kim, and D. Radev, "Creating a neural pedagogical agent by jointly learning to review and assess," 2019, *arXiv:1906.10910*. [Online]. Available: https://arxiv.org/abs/1906.10910

[6] B. C. Gartin and N. L. Murdick, "Idea 2004: The IEP," *Remedial Special Edu.*, vol. 26, no. 6, pp. 327–331, Nov. 2005.

[7] M. M. Elaish, L. Shuib, N. Abdul Ghani, E. Yadegaridehkordi, and M. Alaa, "Mobile learning for English language acquisition: Taxonomy, challenges, and recommendations," *IEEE Access*, vol. 5, pp. 19033–19047, 2017.

[8] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Vis., Graph., Image Process.*, vol. 37, no. 1, pp. 54–115, Jan. 1987.

[9] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 698–713, 1992.

[10] A.-H. Tan, G. A. Carpenter, and S. Grossberg, "Intelligence through interaction: Towards a unified theory for learning," in *Proc. Int. Symp. Neural Netw.* Berlin, Germany: Springer, 2007, pp. 1094–1103.

[11] G.-M. Park, J.-W. Choi, and J.-H. Kim, "Developmental resonance network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1278–1284, Apr. 2019.

[12] J.-Y. Park and J.-H. Kim, "Incremental class learning for hierarchical classification," *IEEE Trans. Cybern.*, vol. 50, no. 1, pp. 178–189, Jan. 2020.

[13] J.-Y. Park and J.-H. Kim, "Online incremental classification resonance network and its application to human-robot interaction," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/tnnls.2019.2920158.

[14] S. Y. Wong, K. S. Yap, C. P. Lim, and E. W. M. Lee, "Hybrid data regression model based on the generalized adaptive resonance theory neural network," *IEEE Access*, vol. 7, pp. 116438–116452, 2019.

[15] N. Masuyama, C. K. Loo, H. Ishibuchi, N. Kubota, Y. Nojima, and Y. Liu, "Topological clustering via adaptive resonance theory with information theoretic learning," *IEEE Access*, vol. 7, pp. 76920–76936, 2019.

[16] G.-M. Park, Y.-H. Yoo, D.-H. Kim, and J.-H. Kim, "Deep ART neural model for biologically inspired episodic memory and its application to task performance of robots," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1786–1799, Jun. 2018.
[17] U.-H. Kim and J.-H. Kim, "A stabilized feedback episodic memory (SF-EM) and home service provision framework for robot and IoT collaboration," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2018.2882921.
[18] J. Nasir, D.-H. Kim, and J.-H. Kim, "ART neural network-based integration of episodic memory and semantic memory for task planning for robots," *Auton. Robot.*, vol. 43, no. 8, pp. 2163–2182, Dec. 2019.
[19] W. Wang, B. Subagdja, A.-H. Tan, and J. A. Starzyk, "Neural modeling of episodic memory: Encoding, retrieval, and forgetting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 10, pp. 1574–1586, Oct. 2012.
[20] W.-H. Lee and J.-H. Kim, "Hierarchical emotional episodic memory for social human robot collaboration," *Auton. Robot.*, vol. 42, no. 5, pp. 1087–1102, Jun. 2018.

WOO-HYUN KIM received the B.S. and M.S. degrees in electrical engineering from KAIST, South Korea, in 2007 and 2009, respectively, where he is currently pursuing the Ph.D. degree. His current research interests include artificial intelligence, machine learning, online tutoring systems, and human and robot interaction.

JONG-HWAN KIM (Fellow, IEEE) received the Ph.D. degree in electronics engineering from Seoul National University, South Korea, in 1987. Since 1988, he has been with the School of Electrical Engineering, KAIST, South Korea, where he is leading the Robot Intelligence Technology Laboratory, as a KT Endowed Chair Professor. He is currently the Director of the KoYoung-KAIST AI Joint Research Center and the Machine Intelligence and Robotics Multi-Sponsored Research and Education Platform. His research interests include intelligence technology, machine intelligence learning, and AI robots. He has authored five books and five edited books, two special journal issues, and around 400 refereed articles in technical journals and conference proceedings.