IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 12, 2021, accepted October 28, 2021, date of publication November 2, 2021, date of current version November 11, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3125060

Tooee: A Novel Scratch Extension for K-12 Big Data and Artificial Intelligence Education Using Text-Based Visual Blocks

YOUNGKI PARK[1] AND YOUHYUN SHIN[2]
[1]Department of Computer Education, Chuncheon National University of Education, Chuncheon, Gangwon-do 24328, Republic of Korea
[2]Department of Computer Science and Engineering, Incheon National University, Incheon 22012, Republic of Korea

Corresponding author: Youhyun Shin (yhshin@inu.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2020R1I1A3068836.

This work involved human subjects or animals in its research. Approval of all ethical and experimental procedures and protocols was granted by the Institutional Review Board of the Chuncheon National University of Education under Application Number 2021-15.

ABSTRACT Many approaches have been proposed to teach the basic concepts of big data and artificial intelligence to K-12 students based on block-based programming languages, such as Scratch. Using these approaches, young students can easily experience big data and artificial intelligence through a drag-and-drop approach. However, it remains difficult for them to perform more complex tasks, such as directly collecting data from the web or exploiting custom-made machine learning algorithms. In this paper, we propose a novel Scratch extension that allows Scratch to communicate with text-based programming languages such as Python and JavaScript using WebSockets. Unlike other Scratch extensions, our extension greatly enhances the extensibility of Scratch given its use of ''text-based visual blocks'' so that messages can be freely exchanged through a minimum number of blocks. In order for students to use these blocks easily, the blocks are designed such that they can be used as if talking with a friend named ''Tooee.'' In order to show how this extension can help students create big data and artificial intelligence programs, we present eight example applications that students can easily implement. These are (1) Weather Forecast, (2) Top 5 Movies in Theaters, (3) COVID-19 Dashboard, (4) Saving Quiz Results to a CSV File, (5) Facial Image Classification, (6) Color Classification, (7) Object Classification, and (8) Handwriting Recognition. Our analyses and experimental results show that Tooee has several advantages over other educational environments.

INDEX TERMS Artificial intelligence education, big data education, K-12, Scratch, Tooee.

I. INTRODUCTION

There are many programming environments that allow K-12 students to learn the basic concepts of big data and artificial intelligence (AI) [1]–[10]. Through these environments, young students can implement basic-level big data and AI programs; for example, students can (1) visualize national cabbage production data [11], (2) perform a sentiment analysis using Twitter data [2], (3) train a machine learning model that distinguishes between a dog and a cat [3], and (4) create a game program using a pre-trained model that recognizes the position of a face [1]. In order to implement these types of programs, students often use block-based programming languages such as Scratch. By using these programming languages, they can implement programs as easily as assembling LEGO blocks by dragging and dropping ''visual blocks.''

However, a limitation of these approaches is that because the provided visual blocks are designed to perform only specific types of tasks, it is not easy for students to create various types of big data and AI programs based on these blocks. For example, when using [1] or [2], while students can acquire Twitter data using ''Twitter blocks,'' it's difficult for them to crawl data from Facebook, Instagram or other websites because the corresponding visual blocks are not provided. As another example, while students can train a machine learning model using ''machine learning blocks,'' changing the corresponding neural network structures based on these types of blocks is challenging.

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

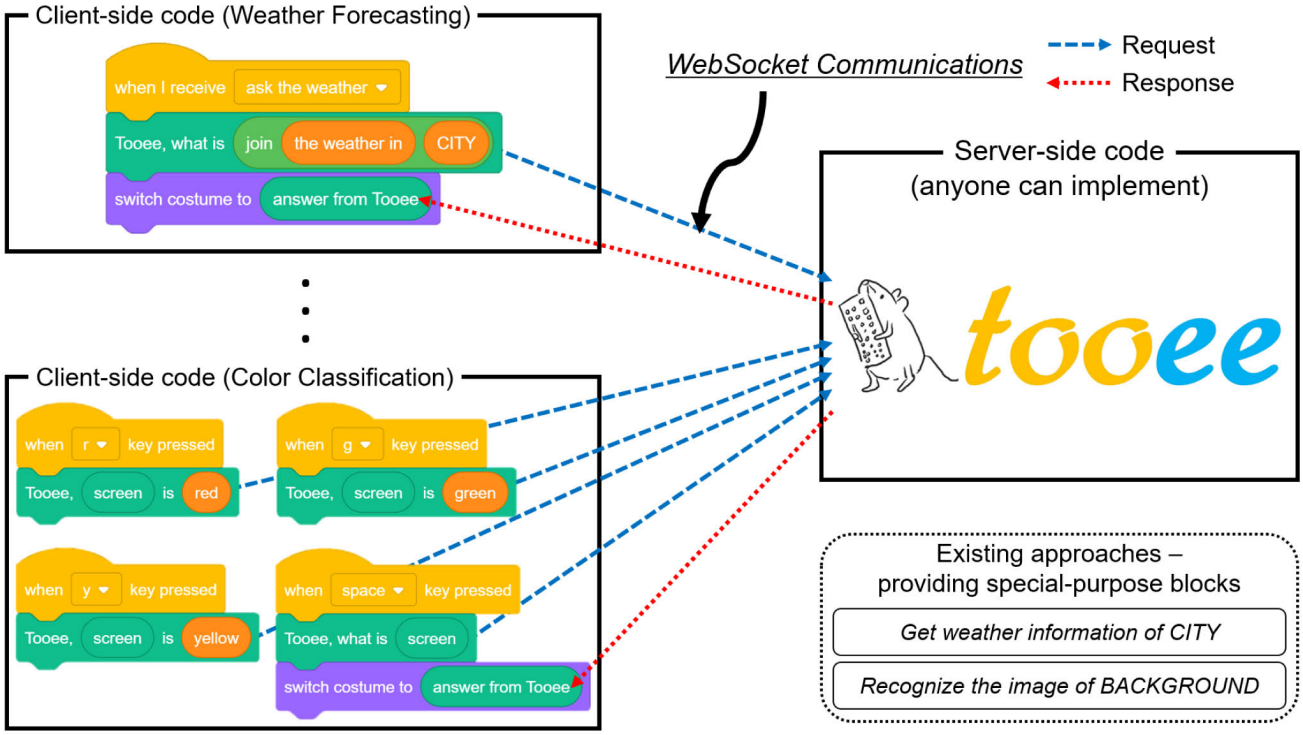


FIGURE 1. An illustrative example of our approach: (1) For Weather Forecast, Scratch asks Tooee for weather information for a specific city, and if Tooee returns "sunny," the sunny icon is then displayed on the screen. (2) For Color Classification, if a user presses the "r" key, Scratch teaches Tooee that the current screen is red. After that, if the user presses the space key when a new screen appears, Scratch asks Tooee what color the current screen is, and if Tooee answers red, a red object appears on the screen.

In this paper, we propose a novel Scratch extension that allows Scratch to communicate with text-based programming languages such as Python and JavaScript using WebSockets. Because it greatly enhances the extensibility of Scratch by using "text-based visual blocks," students can implement various types of big data/artificial intelligence programs based the new blocks. In order for students to use these blocks easily, the blocks are designed such that they can be used as if talking with a friend named "Tooee." Figure 1 shows an illustrative example of our approach. This example highlights how weather forecast and color classification applications can be implemented using the text-based visual blocks that communicate with Tooee, rather than with special-purpose blocks such as "Get weather information of CITY" or "Recognize the image of BACKGROUND."

Here are the research questions of this study:

- RQ1. Can we create a variety of big data/artificial intelligence programs using a block-based programming environment?
- RQ2. What are the advantages of using our proposed extension in K-12 big data/artificial intelligence education?

The main contributions of this paper are as follows:

- We propose a novel Scratch extension that provides text-based visual blocks for the communication with text-based programming languages such as Python and JavaScript (Section III).
- In order to demonstrate how this extension can help K-12 students create their big data and AI applications, we present eight example programs that students can easily implement (Section IV).
- We analyze and discuss the strengths and weaknesses of this extension compared to other Scratch extensions or educational environments (Section V).
- In order to verify that our extension is suitable for big data and artificial intelligence education purposes, we conduct a survey of primary school teachers and analyze the results in detail (Section VI).

II. RELATED WORK

Scratch is the most popular block-based programming language [12], [13]. As of July of 2021, there are more than 74 million Scratch users. The main advantage of this educational programming language is that it allows young students to program easily by dragging and dropping visual blocks. For example, they can create a program that makes a cat move ten pixels when clicking the green flag by simply dragging and dropping the following two visual blocks sequentially onto the cat sprite: (1) the "When Green Flag Clicked" event block and (2) the "Move 10 Steps" command block. Scratch provides most of the basic building blocks of programming, such as events, loops, conditionals, variables, lists, and functions.

However, Scratch rarely provides big data and AI-related visual blocks by default. In order to exploit these types of visual blocks, students must use a modified version of Scratch, such as Machine Learning for Kids [1] or Cognimates [2]. These modifications provide not only additional big data and AI blocks but also their own websites for the training of machine learning models.

Machine Learning for Kids is one of the most popular environments for K-12 AI education based on Scratch. Once students access the Machine Learning for Kids website, they initially must sign up and then log in. After logging in and creating a new project, three menus for implementing AI programs will appear on the screen: (1) Train, (2) Learn & Test and (3) Make. In the Train menu, students can create ''labels'' and prepare their own training data for each. In the Learn & Test menu, they can train a machine learning model based on the data prepared in the Train menu. In the Make menu, students can program using a modified version of Scratch that provides additional blocks for exploiting the model created in the Learn & Test menu. This modified version of Scratch also provides a few big-data-related blocks: (1) a Twitter block to receive the most recent tweets about a specific keyword, (2) and the Wikipedia block for obtaining a Wikipedia article on a specific keyword.

Cognimates is another popular Scratch-based educational environment similar to Machine Learning for Kids. Before using Cognimates, students must register on the Clarifai website and the uClassify website and generate API keys. They can then train machine learning models using the API key on the Cognimates website. The Cognimates website is very similar to that of Machine Learning for Kids: students can create labels, prepare their own training data for each label, train a machine learning model based on the data, and program using the modified version of Scratch that provides additional blocks for exploiting the trained model. Also, Cognimates provides a few big-data-related blocks, including Twitter blocks to obtain the most recent tweets about a specific keyword as well as the latest tweets from a specific person.

A limitation of these modified versions of Scratch is that because their provided visual blocks are designed to perform only specific types of tasks, it is not easy for students to create various types of big data and artificial intelligence programs based on these blocks. For example, if students want to create a program that uses Twitter data, they can implement the program by exploiting *the Twitter blocks* provided by Machine Learning for Kids or Cognimates. However, if students want to create a program that uses data from Instagram or Facebook, there is no way to implement the program until a new block-based programming environment that provides *the Instagram or Facebook blocks* is released. This is a common problem not only for Machine Learning for Kids and Cognimates, but also for other big data/artificial intelligence educational environments such as Teachable Machine [3], DeepScratch [5], Snap! [6], PopBots [7], LearningML [8], Scratch nodes ML [9], Milo [10], and Artificial Intelligence with MIT App Inventor [17]. These limitations inevitably have a negative impact on fostering students' computational thinking [14] skills (e.g., *decomposition*) and their abilities to use big data and artificial intelligence technologies described in ACM CSTA K-12 Computer Science Standards [15] (e.g., *2-DA-08*, *2-DA-09*, or *3B-AP-09*) to some extent.

To address this limitation, we develop a novel Scratch extension named *Tooee*. By using text-based ''conversational'' blocks provided by this extension, we can utilize a variety of functionalities that were only available in text-based programming languages in the block-based programming language. We describe the details of our proposed extension in Section III.

III. TOOEE: A NOVEL SCRATCH EXTENSION

In this section, we propose a novel Scratch extension, referred to as Tooee. First, Section III-A presents the overview of our approach. Second, Section III-B introduces the four design principles involved in creating the extension. Finally, Section III-C introduces five new blocks for K-12 big data and AI education based on the design principles.

A. OVERVIEW OF OUR APPROACH

From a high-level architectural point of view, the Tooee extension provides text-based visual blocks that make it possible to communicate with WebSocket servers. We can use these blocks to not only query the WebSocket server and receive the result, but also give specific instructions to the WebSocket server such as saving a file or training a machine learning model. Because the WebSocket server is usually implemented in a text-based programming language, we can actually utilize the functionalities provided by the text-based programming language through our proposed blocks. Because it is known that synchronous blocks are easier for students to understand than asynchronous blocks when using block-based programming languages [16], all the blocks we propose are implemented to be executed synchronously. For example, if Scratch executes our proposed block that sends a message to the WebSocket server, it waits for the server to reply and then executes the next block. Students who are familiar with text-based programming languages can develop their own WebSocket servers, and those who are not can use public servers (e.g., *http://tooee.org*) or download and run private server programs provided by their teachers.

From an educational point of view, not only do students can create the programs they have envisioned using the blocks, but also they can achieve more diverse learning objectives based on the blocks. Tooee can be regarded as a scaffolding tool. Students can work on tasks within their zone of proximal development, with assistance from Tooee. For example, given the assumption that there are students who are unfamiliar with the concept of web crawling and that a block-based programming language does not provide web crawling blocks, it is unlikely that they will comprehend this concept using such a language on their own. However, because Tooee can provide such a block very easily, students can understand this concept

with this assistance from Tooee. More details are described in the following subsections.

B. DESIGN PRINCIPLES

The proposed Scratch extension is based on the following four design principles: (1) a small number of extra blocks and screen transitions, (2) highly extensible blocks, (3) conversational blocks, and (4) fast and distributable blocks.

1) A SMALL NUMBER OF EXTRA BLOCKS AND SCREEN TRANSITIONS

Block-based programming languages for younger students tend to provide fewer blocks, because dealing with a large number of blocks can be difficult for young students. For example, LEGO WeDo (ages 7+) [18] has 25 visual blocks, Scratch (ages 8-16) [12], [13] has 120 visual blocks, and App Inventor [19], [20] (ages 12-adult) has hundreds of visual blocks. Thus, our aim is to add a minimum number of blocks for K-12 big data and AI education.

Likewise, it can be difficult for young students to use environments that require numerous screen transitions. Therefore, it is necessary to minimize the number of menus provided so that students can program even with fewer screen transitions. This is in line with Scratch Microworlds, which attempts to limit the number of blocks, buttons, and menus [21].

2) HIGHLY EXTENSIBLE BLOCKS

One of the main drawbacks of block-based programming languages (such as Scratch) is that they have a limited number of available functions compared to text-based programming languages (such as Python). That is to say, the types of programs that can be made with block-based programming languages are somewhat limited. For example, while we can easily write programs that access specific files on our local computers using Python, we cannot write the same programs using Scratch because Scratch does not provide any file access blocks. Therefore, our aim is to make highly extensible blocks so to enhance the capabilities of Scratch.

3) CONVERSATIONAL BLOCKS

In order to make easy-to-use blocks, we propose new blocks that are ''conversational.'' We define a conversational block as a block that resembles conversations with friends. For example, in order to retrieve weather information from the web, we will use the block ''Tooee, what is [the weather]?'' instead of the ''get weather information'' block. Conversational blocks have an advantage in that students can easily learn how to use them by executing these blocks and checking the results.

Our conversational blocks should supports follow-up questions. That is to say, our ''conversations'' can continue indefinitely, and when executing a conversational block, it can be affected by the previously executed conversational block. This feature can make programming easier when similar instructions must be executed consecutively. For example, if a student wants to obtain a classification result and its corresponding confidence value, the two related types of information can easily be obtained with our conversational blocks.

4) FAST AND DISTRIBUTABLE BLOCKS

Our aim is to enable students to use the proposed blocks not only for learning the basic concepts of big data and AI but also for creating practical applications with these tools. In order to achieve this aim, the proposed blocks should be executed as quickly as possible. Because even the slowest Scratch block is executed within one second, our aim is to create blocks that run in a maximum of one second. The proposed blocks should also be ''distributable,'' which means that after students create their applications based on the blocks, they should easily be able to convert the applications into distributable files (e.g., HTML or EXE).

C. BLOCK INTERFACES

Figure 2 shows the proposed blocks. When a user clicks the ''Choose an extension'' menu in Scratch, the Tooee extension button appears on the screen, as shown in the figure on the left of Figure 2. Subsequently, when the Tooee extension button is clicked, the five blocks shown in the figure on the right of Figure 2 are added to Scratch. In this subsection, we describe in detail how each block works. Note that in Section IV, we will explain how to make real-world applications using these blocks.

1) CONNECTION BLOCK

The Tooee extension provides the ''connection block,'' which establishes a connection between Scratch (a Web browser) and a WebSocket server. The interface of this block is as follows:

- *Set Tooee's address to (localhost:9998)*

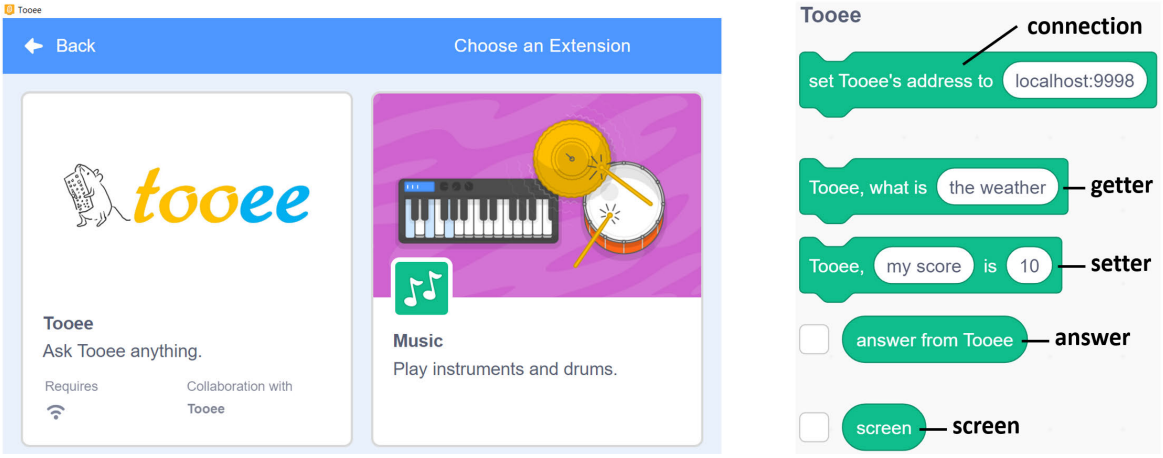
Here, we abstract the WebSocket server into the virtual friend ''Tooee.'' Therefore, setting Tooee's address indicates establishing a connection between Scratch and the WebSocket server. Once this block is executed, the connection is established and remains until the program is terminated or a new connection is made. The default address ''localhost:9998'' is specified in advance inside the parentheses. However, students can easily change the address by either typing another value or dragging and dropping another block into the parentheses. In other words, they can set Tooee's address as either a public server address (e.g., *http://tooee.org*) or one of their private server addresses. Example server programs will be provided on the Tooee website (*http://tooee.org*).

2) GETTER BLOCK AND ANSWER BLOCK

The Tooee extension provides a ''getter block,'' which requests information from the WebSocket server. The interface of this block is shown below.

- *Tooee, what is (the weather)*

Here, we abstract the WebSocket server into the virtual friend ''Tooee.'' Therefore, asking Tooee questions equates to requesting information to the WebSocket server. The default



(a) (b)

FIGURE 2. Novel Scratch extension *Tooee*. (a) If a user clicks the Tooee extension in the "Choose an Extension" menu, (b) five blocks are added to Scratch. The added blocks consist of a "connection" block, a "getter" block, a "setter" block, an "answer" block and a "screen" block.

request is specified in advance inside the parentheses. However, students can easily change the request by either typing another request or dragging and dropping another block into the parentheses. When this block is executed, the WebSocket server receives the request string (e.g., "the weather"), processes the request, and stores the result in the following "answer block."

- *answer from Tooee*

Here is an example scenario of how the getter block and the answer block can be used together. (1) First, Scratch executes the block "Tooee, what is (the weather)," and the string "the weather" is sent to the WebSocket server. (2) After that, the WebSocket server fetches the weather information from the Meteorological Office website and stores the "sunny" string in the "answer from Tooee" block. (3) Finally, Scratch checks the value contained in the answer block and shows a picture of a sun on the screen.

3) SETTER BLOCK AND ANSWER BLOCK

The Tooee extension provides the "setter block," which also sends a message to the WebSocket server. The block interface of this block is as follows:

- *Tooee, (my score) is (10)*

The setter block is identical to the getter block, except that the setter block sends the concatenated string of (1) the value in the first set of parentheses, (2) the string [tooeeDelimeter], and (3) the value in the second set of parentheses to the WebSocket server. The WebSocket server can then distinguish whether the received request was generated by the getter block or the setter block. Note that all setter blocks can be implemented with getter blocks. However, students can write more intuitive code by using both a getter block (to query the virtual friend Tooee) and a setter block (to inform Tooee of new information).

Here is an example scenario of how the setter block can be used. (1) First, Scratch executes the block "Tooee, (my score) is (10)," and the string "my score [tooeeDelimeter] 10" is sent to the WebSocket server. (2) The WebSocket server then writes "my score, 10" into a "score.csv" file. (3) If an error occurs when saving to the file, the server stores "error" in the "answer from Tooee" block. Otherwise, it stores "success" in the answer block. (4) Finally, Scratch checks the value contained in the answer block. If the value is "success," Scratch then shows a message saying that the player's score has been successfully saved to the file.

4) SCREEN BLOCK

The Tooee extension provides a "screen block," which contains Base64-encoded image data pertaining to the current screen. The block interface of this block is as follows:

- *screen*

This block can be used with either the getter block or the setter block. For example, when students want to teach Tooee that the picture displayed on the screen is a cat, the screen block and the setter block can be used together. As another example, when they want to ask Tooee what animal the picture displayed on the current screen represents, the screen block and the getter block can be used together.

IV. EXAMPLE APPLICATIONS FOR K-12 EDUCATION

In this section, we present eight example applications using the Tooee extension. Among them, the first four programs are related to K-12 big data education (Weather Forecast, Top 5 Movies in Theaters, COVID-19 Dashboard, and Saving Quiz Results to a CSV File), and the last four programs are related to K-12 AI education (Facial Image Classification, Color Classification, Object Classification, and Handwriting Recognition). Table 1 shows how each program was

TABLE 1. Eight example programs for K-12 big data and K-12 AI education. This table briefly describes how Scratch (front-end development) and Tooee (back-end development) were used together to implement each program. Further details are described in Section IV.

Area	Program	Scratch (Front-end Development)	Tooee (Back-end Development)
K-12 Big Data	Weather Forecast	Display tomorrow's weather information at the location selected by the user.	Fetch weather information (weather conditions, lowest temperature, highest temperature) from the Meteorological Agency website.
	Top 5 Movies in Theaters	Show the information about movies with the highest number of visitors per day.	Fetch movie information (movie title, daily number of visitors, cumulative number of visitors) from the Box-office Information System website.
	COVID-19 Dashboard	Display the number of confirmed COVID-19 cases yesterday.	Extract the number of confirmed COVID-19 cases from the Ministry of Health and Welfare website.
	Saving Quiz Results to a CSV File	Present multiple-choice questions and allow participants to select one answer choice for each question.	Save the answer choices selected by the participants and their final scores into a CSV file.
K-12 AI	Facial Image Classification	Show a "Hello!" message when an alien is shown on the screen, and hide the message otherwise.	Train a machine learning model to be able to distinguish whether or not an alien is shown on the screen.
	Color Classification	Make a hot-air balloon change color depending on the color of the object displayed on the screen.	Train a machine learning model to be able to classify a red object, a blue object, and a yellow object.
	Object Classification	After classifying an object, display the predicted class and its confidence value graphically.	Train a machine learning model to be able to both classify a cat and a dog, and calculate the confidence value of the prediction.
	Handwriting Recognition	Display the recognition result of a handwritten digit if the confidence value of the prediction is high enough, and display a panda that looks very curious otherwise.	Train a machine learning model to be able to both classify handwritten digits from 1 to 4, and calculate the confidence value of the prediction.

implemented in terms of front-end development (Scratch) and back-end development (Tooee).

Note that back-end development is not mandatory for students. There are two options for students who are not familiar with back-end programming languages. (1) The first option involves simply setting Tooee's address to a publicly available WebSocket server, such as *tooee.org*. (2) The second option is to run one of the existing WebSocket server programs that can be downloaded from the Tooee website (*http://tooee.org*), and set Tooee's address to the corresponding IP address. If students choose one of these two options, then they can focus on front-end development.

Some of the relevant CSTA learning objectives [15] for the first four programs are 1A-DA-05, 1A-DA-06, 2-AP-16, and 3A-AP-17. Those for the last four programs are 1B-DA-06, 2-DA-08, 2-DA-09, 3B-AP-08, and 3B-AP-09.

A. WEATHER FORECAST

The first example application for K-12 big data education is *Weather Forecast*. Example screenshots are shown on the left in Figure 3. In this program, there are four locations in South Korea that can be selected by a user: CNN (short for Chuncheon), SEL (short for Seoul), ICN (short for Incheon), and DJN (short for Daejeon). If the user selects one of these locations, the weather information (weather condition, lowest temperature, highest temperature) of the selected location will then be displayed on the screen. More specifically, the program works as follows:

- When Green Flag is clicked (when the program starts), the connection between Scratch and Tooee is established by the execution of the connection block.
- When a user clicks a specific location, Scratch broadcasts three messages (''ask the weather,'' ''ask the lowest temperature,'' and ''ask the highest temperature'').
- When one of these messages is received, Scratch asks Tooee for the corresponding weather information about the selected city and changes the corresponding weather icon based on the answer from Tooee.

The main Scratch code is demonstrated on the right side of Figure 3. Here, we assume that the orange-colored variable blocks are pre-defined by a teacher, as follows:

- The ''MY COMPUTER'' block has a WebSocket server address (e.g., *tooee.org*).
- The ''CITY'' block has the location selected by the user (e.g., *CCN*).
- The variable blocks in lowercase contain content matching their block names (e.g., the ''the weather in'' block has the string *the weather in*).

The back-end program can easily be implemented; when it receives a request, it fetches the weather information (weather condition, lowest temperature, highest temperature) from the Meteorological Agency website and returns the corresponding information.

B. TOP 5 MOVIES IN THEATERS

The second example application is *Top 5 Movies in Theaters*. An example screenshot is shown on the left in Figure 4. In this program, there are five numbers (1 to 5) that can be selected by a user. When a user clicks on one of these numbers (i.e., k), the information on the movie with the k^{th} highest number of visitors per day is displayed. For example, when

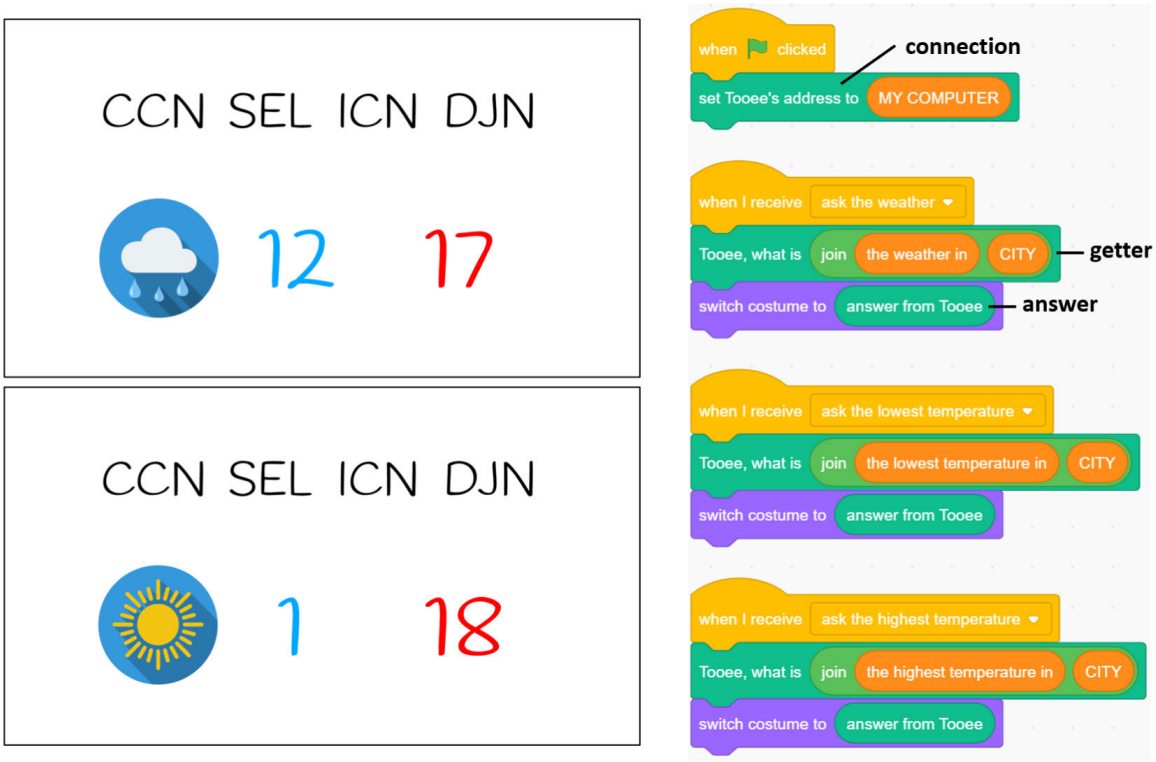


FIGURE 3. Weather Forecast example for K-12 big data education. When the program starts, the connection between Scratch and Tooee is established. When a user clicks on a specific location name among Chuncheon (CCN for short), Seoul (SEL for short), Incheon (ICN for short), or Daejeon (DJN for short), Scratch asks Tooee for the weather condition, lowest temperature, and highest temperature, and changes the weather icons according to Tooee's answer.

the user clicks on "5" on 31 March 2021, the program shows that the movie title is *Minari*, the number of visitors per day is fewer than *100,000*, and the cumulative number of visitors is fewer than *1,000,000*.

Note that the main Scratch code of this application (shown on the right in Figure 4) is very similar to that of the Weather Forecast application. The connection block and the getter blocks are used in the same way. The differences are that the "say" block is used in addition to the "switch costume to" block, that Scratch receives different messages, and that different orange-colored variable blocks are used.

The back-end program can also be easily implemented; when it receives a request, it fetches the movie information from the Box-office Information System website and returns the corresponding information.

C. COVID-19 DASHBOARD

The third example application is *COVID-19 Dashboard*. Example screenshots are shown on the left in Figure 5. In this program, there is a virus-shaped character that can be clicked by a user. If the user clicks on this, the character changes its color and the number of confirmed COVID-19 cases is displayed on the screen. More specifically, the program works as follows:

- When Green Flag clicked, a connection between Scratch and Tooee is established by executing the connection block.
- When a user clicks the virus-shaped character, Scratch broadcasts the "ask Tooee" message. When this message is received, Scratch asks Tooee for the number of confirmed COVID-19 cases, after which Scratch selects and displays the appropriate costume using the answer block.

The main Scratch code is demonstrated on the right in Figure 5. Here, similar to the Weather Forecast program, we assume that the orange-colored variable blocks are predefined by a teacher. Note the back-end program can also be easily implemented; when it receives a request, it extracts the number of confirmed COVID-19 cases from the Ministry of Health and Welfare website and returns this number to Scratch.

D. SAVING QUIZ RESULTS TO A CSV FILE

The last example application for K-12 big data education is *Saving Quiz Results to a CSV File*. In contrast to the previous applications, this program exploits the setter block instead of the getter block. An example screenshot is shown on the top left in Figure 6. Note that we mimic the popular quiz program *Kahoot!*. When the program starts, multiple-choice questions are shown on the screen. Whenever a user selects an answer choice, the selection is recorded in a CSV file, as shown on the bottom-left area of Figure 6. More specifically, the program works as follows:

- When Green Flag clicked, (1) a connection between Scratch and Tooee is established by the execution of the

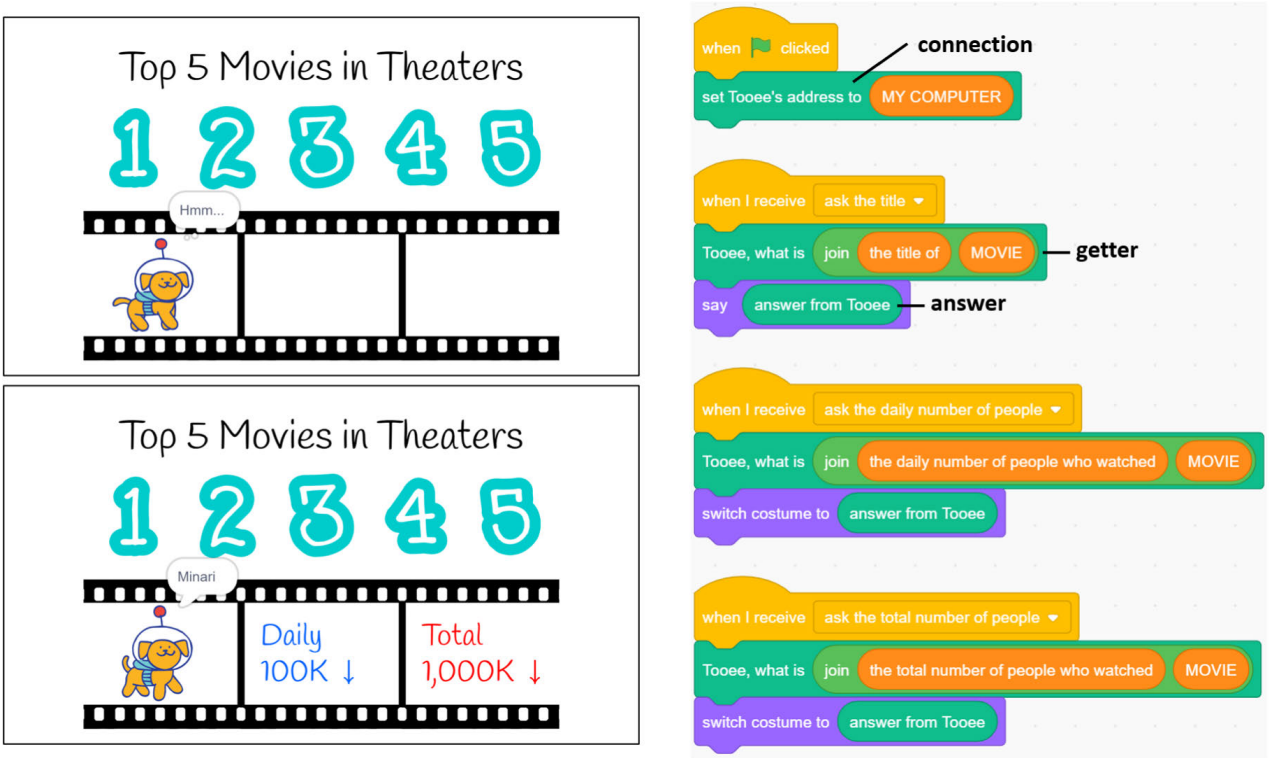


FIGURE 4. Top 5 Movies in Theaters example for K-12 big data education. When the program starts, a connection between Scratch and Tooee is established. When a user clicks a number from 1 to 5 (i.e., k), Scratch asks Tooee for information about the movie with the k^{th} highest number of visitors per day, and the Tooee's answers (movie title, number of visitors per day, and cumulative number of visitors) are displayed on the screen.

connection block, and (2) all of the elements in the list named "RESULT LIST" are deleted.

- Whenever a user clicks on one of the answer choices, Scratch (1) stores the user's choice in the "MY ANSWER" block, (2) updates the user's score in the "MY SCORE" block, and (3) broadcasts the message "a button clicked." When this message is received, the content of the "MY ANSWER" block is added to the "RESULT LIST."
- Once the user has answered all of the questions, the message "save the results" is broadcast. When this message is received, the user's score is added to the "RESULT LIST" and Scratch asks Tooee to save the elements of the list into a CSV file.

The main Scratch code is demonstrated on the right in Figure 6. Here, we assume that the "MY COMPUTER" block has a WebSocket server address (e.g., *tooee.org*) and that the "what you need to save in the CSV file" block has content matching its block name. Note that the back-end program is very easy to implement; when it receives a list, the elements of the list are saved in the CSV file with comma delimited.

E. FACIAL IMAGE CLASSIFICATION

The first example application for K-12 AI education is *Facial Image Classification*. The example screenshots are shown on the left of Figure 7. When the program starts, an uncolored background appears and a webcam is automatically turned on. If the alien's face is captured by the webcam, a welcome message is displayed on the screen. If the face is not classified as an alien, the welcome message disappears.

One example Scratch code to implement this application is demonstrated on the right of Figure 7. When Green Flag clicked (when the program starts), the connection between Scratch and Tooee is established by using the connection block, and repeatedly execute the following statements: (1) Scratch sends the current screen to Tooee using the getter block and the screen block. (2) If Tooee returns the string *alien*, then the welcome message appears on the screen. Otherwise (if Tooee returns the string *not alien*), the message disappears. Here, we assume that the orange-colored variable blocks are pre-defined by a teacher in the same way as the other example applications.

Note that the students should train the machine learning model before Green Flag clicked by pressing the *n* and *y* keys. In this program, when *n* key pressed, both the current screen and the string *not alien* are sent to Tooee. After that, Tooee will learn that this image belongs to the class *not alien* after receiving this information. Similarly, when *y* key pressed, both the current screen and the string *alien* are sent to Tooee for training the class *alien*.

One example of the back-end program can be implemented as follows: (1) when both Base64-encoded image data and

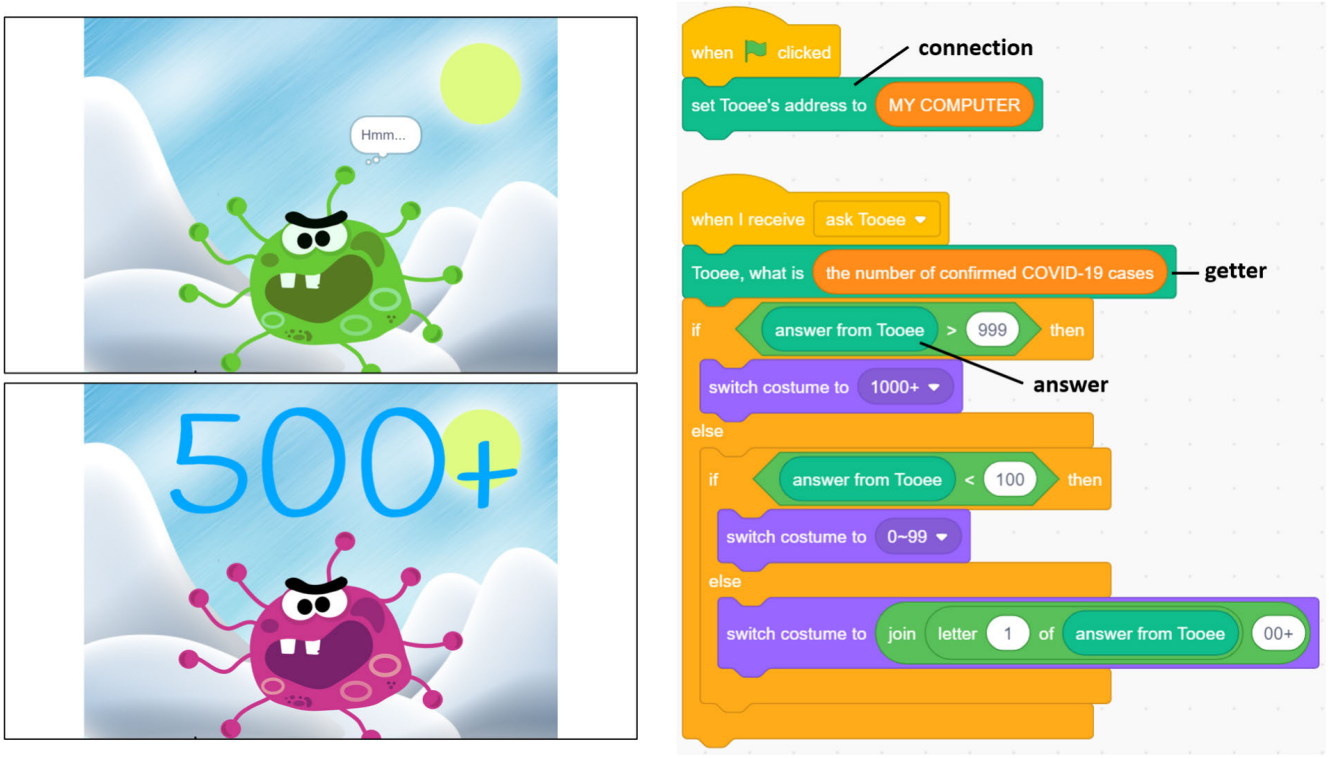


FIGURE 5. COVID-19 dashboard example for K-12 big data education. When the program starts, a connection between Scratch and Tooee is established. When a user clicks on the virus-shaped character, Scratch asks Tooee for the number of confirmed COVID-19 cases, and the Tooee's answer is displayed on the screen.

a class label are received, the back-end program saves the image data in a subdirectory named the class label. (2) When only an Base64-encoded image data are received without a label, the back-end program trains the saved data that has not been trained yet (if they exists), predicts the class label of the received image and returns it to the Scratch program.

F. COLOR CLASSIFICATION

The second example application for K-12 AI education is *Color Classification*. Example screenshots are shown on the left in Figure 8. In this example, there is an amusement park and a hot-air balloon. When a user shows a colored object on the webcam, a hot-air balloon of the same color appears on the screen.

A Scratch code sample to implement this application is demonstrated on the right in Figure 8. When Green Flag is clicked (when the program starts), a connection between Scratch and Tooee is established using a connection block, and the current screen is repeatedly sent to Tooee. If Tooee returns the string *red*, *blue*, or *yellow*, the Scratch program then changes the image of the hot-air balloon to that of the corresponding color.

As with the facial image classification program, users must train a machine learning model. When a user presses *r*, *g*, or *y*, Scratch sends both the Base64-encoded image data and the corresponding label, and Tooee saves the received data to train the model. Note that the back-end program can be written identically to that in facial image classification.

G. OBJECT CLASSIFICATION

The third example applications for K-12 AI education is Object Classification. This application classifies whether the image on the screen is a cat or a dog. In order to train a cat or a dog based on the current screen, users must press the *c* key or the *d* key, respectively. Example screenshots are shown on the left in Figure 9. The screenshot in the upper left shows that the current image on the screen is classified as a cat with a confidence value of 59%. Similarly, the screenshot in the lower left shows that the image currently on the screen is a dog, and the corresponding confidence value is 66%. Note that both the classification result and its confidence value are visually displayed.

One example Scratch code to implement this application is demonstrated on the right in Figure 9. The code is similar to that of Facial Image Classification or Color Classification, except that it not only asks Tooee for the classification result, but also asks for the confidence value as a follow-up question. Therefore, the back-end program of this application should support this type of follow-up question.

H. HANDWRITING RECOGNITION

The fourth example applications for K-12 AI education is *Handwriting Recognition*, and example screenshots are shown on the left in Figure 10. In this application, users can freely write a number using their mouse controller. While a user is writing, the handwriting is classified as 1, 2, 3 or 4 in real time. Note that if the confidence value is not high enough,

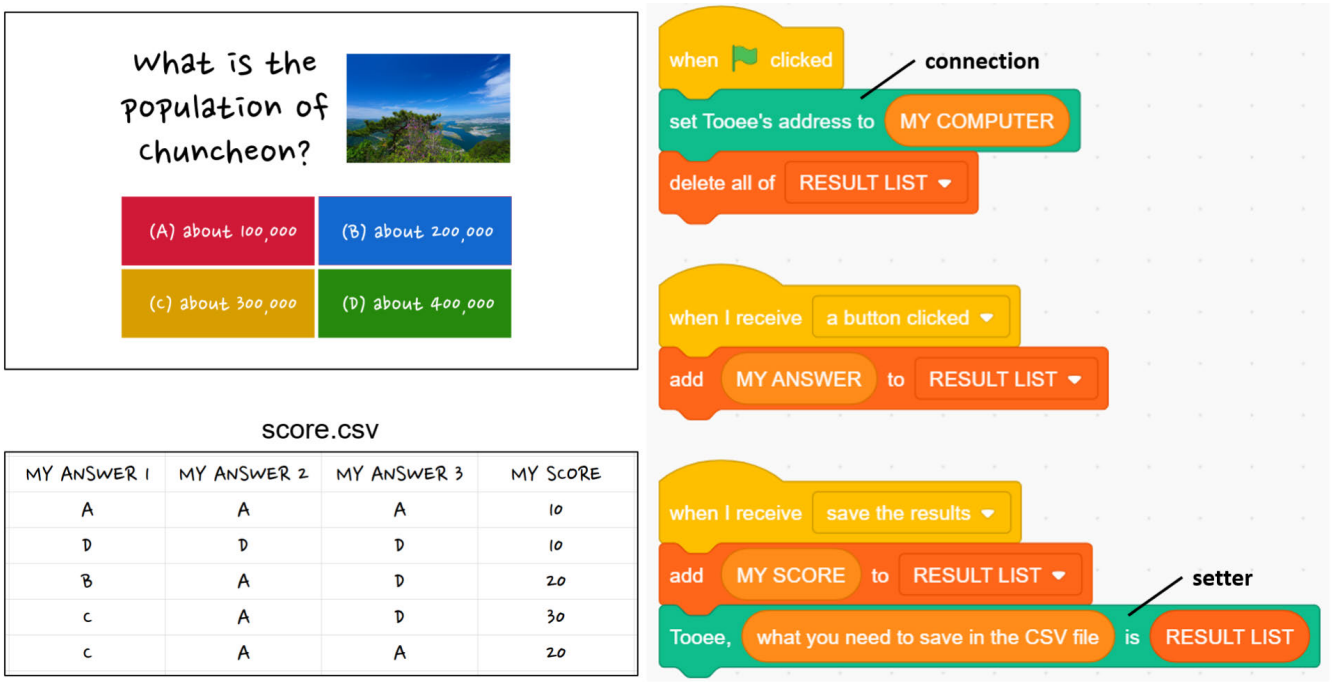


FIGURE 6. Saving quiz results to a CSV file example for K-12 big data education. When the program starts, a connection between Scratch and Tooee is established, an empty list called the "RESULT LIST" is prepared, and multiple-choice questions are shown on the screen. Whenever a user selects an answer choice, the selection is recorded in the list. When the program ends, Scratch adds the final score in the list and sends the list to Tooee, and Tooee saves the elements of the list into a CSV file.

the application shows an image on the screen indicating that it did not recognize the number drawn by the user.

One example Scratch code to implement this application is demonstrated on the right in Figure 10. The code is very similar to that of the object classification example; it initially asks about the class label corresponding to the current screen, after which it asks for the corresponding confidence value as a follow-up question. The major difference from the previous example is that when the confidence value is not high enough, the program shows that the handwriting is not recognized instead of visualizing the confidence value. As with the other AI examples, users can train their own handwritten digits by pressing the *1*, *2*, *3* or *4* key.

V. ANALYSIS

In this section, we analyze this extension in terms of its code complexity, usability, performance, extensibility, and deployability. Experimental results based on teacher surveys will be presented in Section VI.

A. CODE COMPLEXITY

One of the most popular code complexity metrics for Scratch codes is *Dr. Scratch* [22]–[24], which measures seven code complexity areas: *abstraction and problem decomposition (ABS)*, *parallelism (PAR)*, *logical thinking (LOG)*, *synchronization (SYN)*, *flow control (FLO)*, *user interactivity (USE)*, and *data representation (DAT)*. Here, as more difficult-to-use blocks or design patterns are used, a higher code complexity value is measured by Dr. Scratch (called the Dr. Scratch score). For example, if we use "My Blocks," which defines a new function, the ABS score is at least two. As another example, if we use "two scripts on green flag," the PAR score is then at least one, as in this case, multiple code blocks should be executed at the same time, which can complicate the Scratch code [16].

The Dr. Scratch score can be increased by using the Tooee extension. (1) If we use the connection block, the getter block, or the setter block, the Scratch program must then communicate with the back-end program. Because this communication is similar to the communication between sprites (objects) using the "broadcast" block, we can say that the SYN score is at least two points if we use one of these blocks. (2) Also, we often use the *forever* Scratch block (an infinite loop) together with the getter block. Because the FLO score is at least two points when using the forever block, the use of the getter block can lead to an increase in the FLO score. (3) Similarly, the use of the setter block can lead to an increase in the USE score because we often use the *key pressed* event block (two points in the USE area) in order to train a machine learning model. (4) Because both the answer block and the screen block can be considered as variables, the use of these blocks can be considered to obtain a DAT score of at least two points. (5) Finally, when a Scratch project exploits the screen block, the project usually gains a USE score of three points, as the block is often used together with video blocks.

Table 2 shows the Dr. Scratch score for each example application introduced in Section IV. In this table, the average Dr. Scratch score of the eight projects is *13.625*. Because the

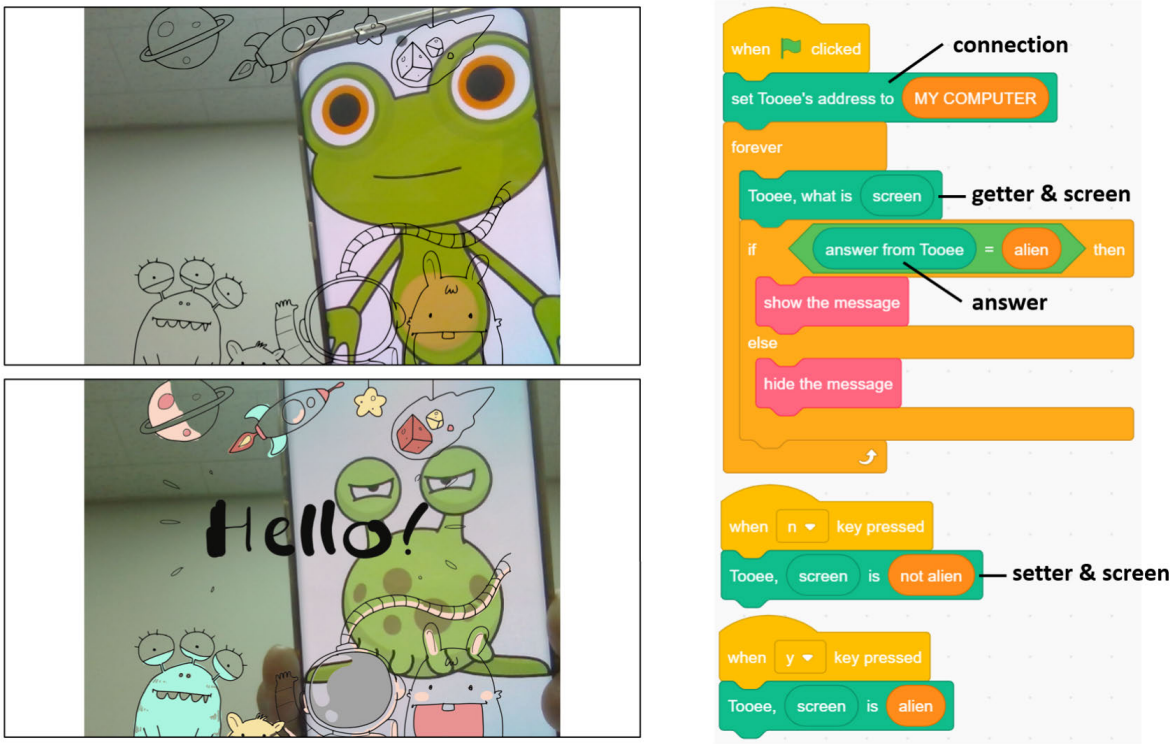


FIGURE 7. Facial image classification example for K-12 AI education. When the program starts, a connection between Scratch and Tooee is established, and Scratch repeatedly sends the current screen to Tooee. If the response from Tooee is "alien," the message "Hello!" is then displayed on the screen, and other alien images are turned into color images. Otherwise, there will be no change on the screen. A user can train Tooee by pressing the "y" or "n" key.

average score does not different greatly from those of other projects made by Scratch beginners [23], young students will easily be able to create and understand the example applications. According to this table, the easiest project to implement is *Color Classification* (nine points in total). Because the most difficult part of implementing this project involves the use of video blocks, students who have experience using video blocks through an introductory Scratch class can easily implement this program. On the other hand, the most difficult projects to implement are Object Classification and Saving Quiz Results to a CSV file (18 and 16 points in total, respectively). For *Object Classification*, the "display the results by using" block, which performs data visualization, raises the Dr. Scratch score significantly. However, if a teacher provides that block in advance, students can easily implement this application. Similarly, for *Saving Quiz Results to a CSV File*, if students program using only variables without using lists, the Scratch code will be much easier to implement.

B. USABILITY

1) A NUMBER OF EXTRA BLOCKS AND MENUS

As noted in Section III-B, one of our design principles is to provide a small number of extra blocks. In reality, if we do not count the number of pre-trained model blocks and pre-defined variable blocks, there are 5, 16, and 23 extra blocks for *Tooee*, *Machine Learning for Kids*, and *Cognimates*, respectively. Given the low number of Tooee blocks, Tooee has an advantage in that students can easily become familiar with its blocks. On the other hand, because many tasks can be done with a small number of blocks, it may take somewhat longer for students to learn how to use each block.

Recall also that one of our design principles is to reduce the number of extra menus. While Machine Learning for Kids and Cognimates provide additional menus for the training and testing of machine learning models, Tooee does not provide a separate menu for big data and AI-related programming. That is to say, Tooee users can create various types of programs using only the code editor. Therefore, Machine Learning for Kids and Cognimates are suitable for users who want to check how AI learns in a separate menu, and Tooee is suitable for users who find it difficult to switch screens frequently.

2) REGISTRATION, LOGIN, AND API KEYS

When using Tooee, a user can exploit big data and AI-related blocks by either using the address of a public WebSocket server or by executing a WebSocket server program. On the other hand, when using Machine Learning for Kids or Cognimates, a user must register and log in as a member, and must enter API keys when required. Thus, we can say that Tooee is a relatively easy extension for students to start big data and AI programming because little preparation is required.

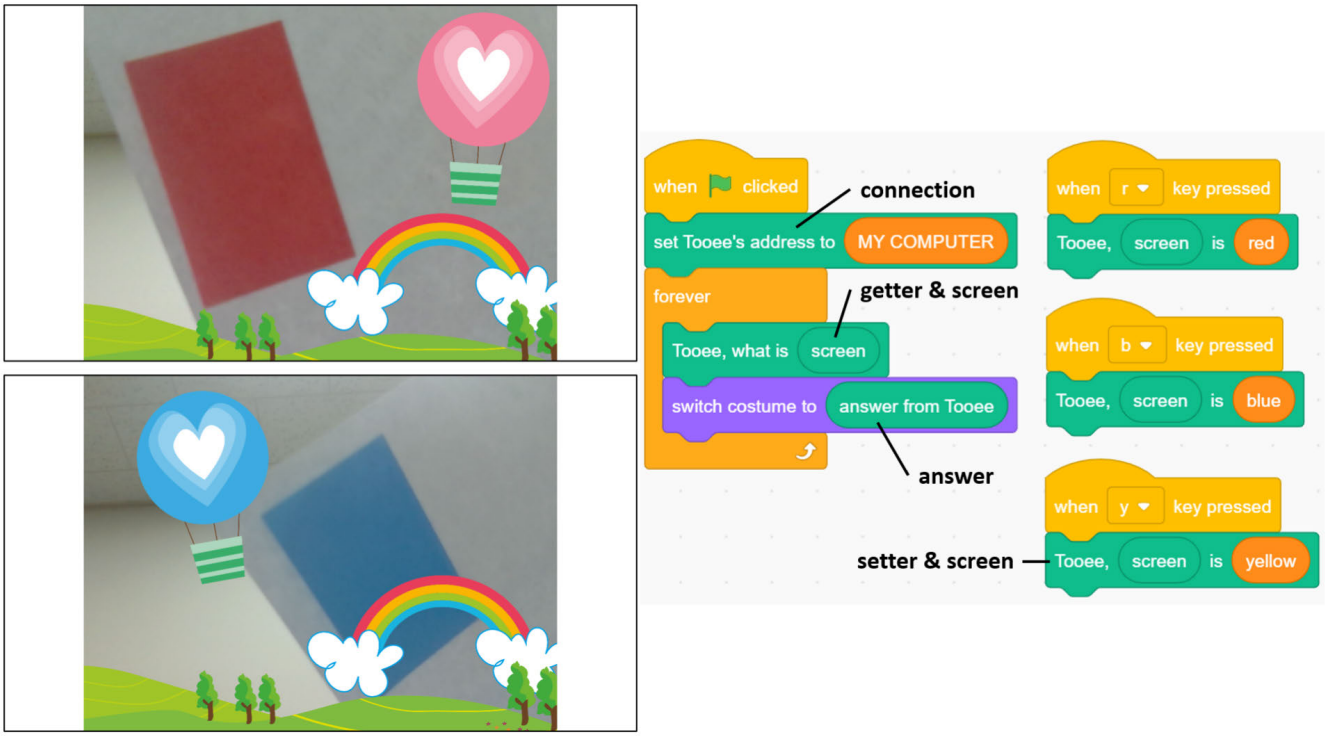


FIGURE 8. Color classification example for K-12 AI education. When the program starts, a connection between Scratch and Tooee is established, and Scratch repeatedly sends the current screen to Tooee. If the response from Tooee is "red," the red hot-air balloon is then shown on the screen. The same is also true for blue and yellow hot-air balloons. In order to train Tooee to classify colors, it is necessary to press the "r," "b" or "y" keys on the red, blue, or yellow screens, respectively.

However, it could be relatively easier for users to manage multiple AI projects when using Machine Learning for Kids or Cognimates, because their user accounts are maintained.

C. PERFORMANCE

As stated in Section III-B, one of our design principles is to present visual blocks that should be executed as quickly as possible in order to make it easy to create practical applications. Because Scratch and Tooee communicate with each other using WebSockets, in this subsection, we measure the WebSocket communication speed to show that our proposed block executes as fast as we want.

For big-data-related applications, Scratch usually sends a short string to Tooee (e.g., "what is the weather in Chuncheon") and Tooee returns a short string (e.g., "sunny") to Scratch. On the other hand, for AI-related applications, Scratch frequently sends an entire screen (approximately 340KB) to Tooee, and Tooee returns a short string (e.g., "cat") to Scratch. Thus, we assume that if the process of sending 340 kilobytes of data and receiving ten bytes of data can be done quickly, students will be able to make both big data and AI related fast running programs. When we use a personal laptop computer (2.8 GHz Intel Core i7 processor) and when communication is with localhost, the sending (340KB) and receiving (10B) process takes approximately 0.067 seconds.

D. EXTENSIBILITY

One major strength of Tooee compared to other Scratch extensions is its extensibility. Because Tooee can almost fully exploit the capabilities of text-based programming languages such as Python and JavaScript, there are virtually no restrictions on making practical applications. Two typical examples demonstrating the high extensibility of Tooee are as follows: (1) Using Tooee, students can implement all of the big data example applications introduced by Machine Learning for Kids or Cognimates. (2) On the other hand, Using Machine Learning for Kids or Cognimates, students cannot implement any of the big data example applications introduced in Section IV.

Tooee is also very extensible even when making AI-related applications. For example, while we can use a PyTorch model when using Tooee, we cannot exploit the same model using Machine Learning for Kids or Cognimates. As another example, while the amount of image data that can be trained is not limited when using Tooee, there are restrictions on the number of training images when using Cognimates (maximum 20 images per class) or Machine Learning for Kids (maximum 100 images).

E. DEPLOYABILITY

One of the most important factors when creating practical programs is the ability to distribute the programs

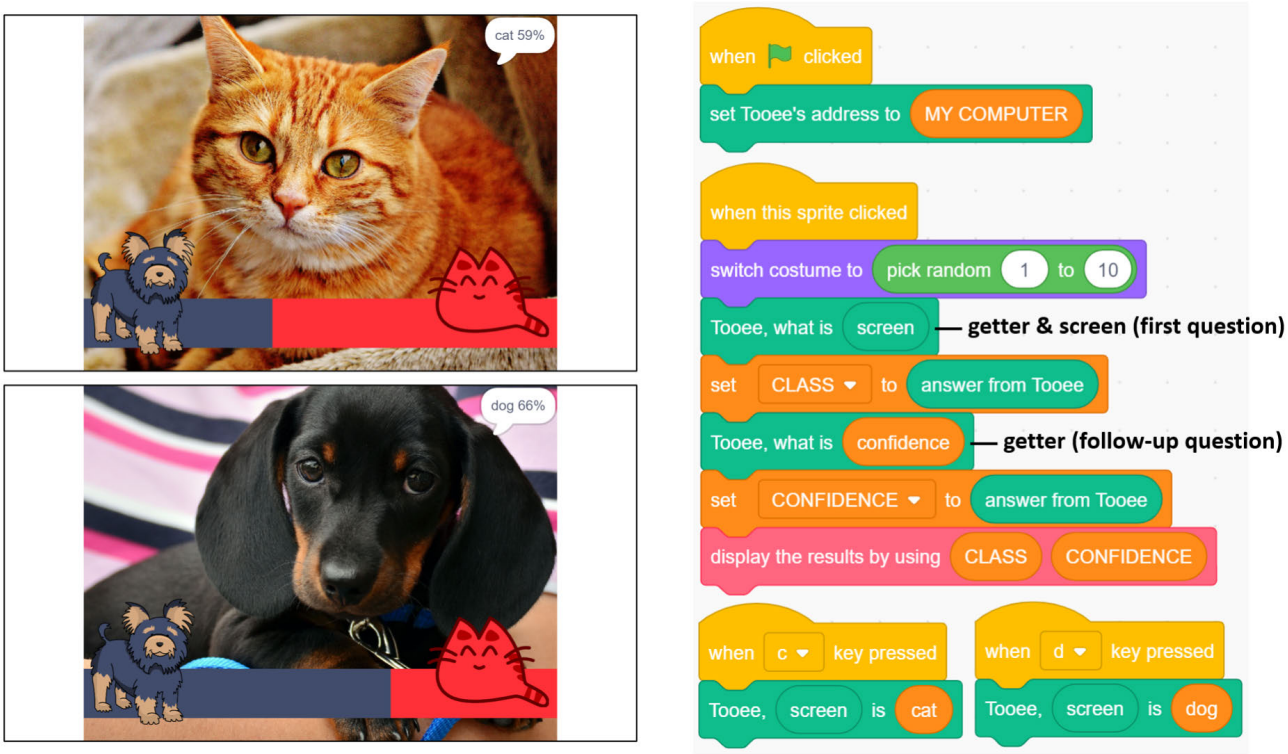


FIGURE 9. Object Classification example for K-12 AI education. When the program starts, a connection between Scratch and Tooee is established. When a user clicks on a background image, the image changes randomly, and the program determines whether this image is a cat or a dog based on the confidence value calculated by Tooee. The user can train Tooee by pressing the "c" or "d" key.

we implement. We can say that Tooee is more *deployable* than other big data or AI-related Scratch extensions for two reasons: (1) in order to share a project, we need to copy the corresponding Scratch file (.SB3) when using Machine Learning for Kids or Cognimates. On the other hand, when using Tooee, we can share an HTML or EXE file converted from a SB3 file. (2) Machine Learning for Kids has a limitation on the number of projects that can be maintained (maximum of three), and Cognimates has a limitation on the number of free machine learning operations per month. On the other hand, when using Tooee, there are no explicit limitations on the number of projects or operations. The limitation is determined by how many operations our WebSocket server can handle.

VI. EXPERIMENTS

In this section, we analyze how effective our proposed extension is in K-12 big data and artificial intelligence education by conducting a teacher survey. Section VI-A describes how the survey was conducted, and Section VI-B demonstrates and analyzes the survey results.

A. EXPERIMENTAL SETUP

To test whether this educational environment (Tooee) is effective for K-12 big data and artificial intelligence education, we conducted an IRB approved online survey involving 15 primary school teachers. Figure 11 summarizes the teachers' teaching and programming backgrounds. (1) All teachers except one had more than five years of teaching experience and had at least one year of teaching experience in computer science. (2) While most of the teachers had a considerable amount of experience with block-based programming languages such as Scratch, they did not have much experience with text-based programming languages such as Python. (3) All teachers except one already had experience using environments for big data/artificial intelligence education, such as Machine Learning for Kids and Cognimates.

First, we asked the teachers to evaluate how effective "existing environments" for K-12 big data/artificial intelligence education are. Then we asked teachers to watch and follow a ten-minute video tutorial showing the process of (1) installing "Tooee," (2) implementing the eight example applications presented in Section IV using Tooee, and (3) distributing the implemented program as an HTML file. Finally, we asked teachers to evaluate how effective Tooee was for big data/artificial intelligence education purposes. These questions aim to measure code complexity, usability, performance, extensibility, and deployability. The detailed survey questions are described in Table 3. For each question, teachers responded on a five-point Likert scale. Here, a score closer to 5 is positive ("yes"), while a score closer to 1 is negative ("no").

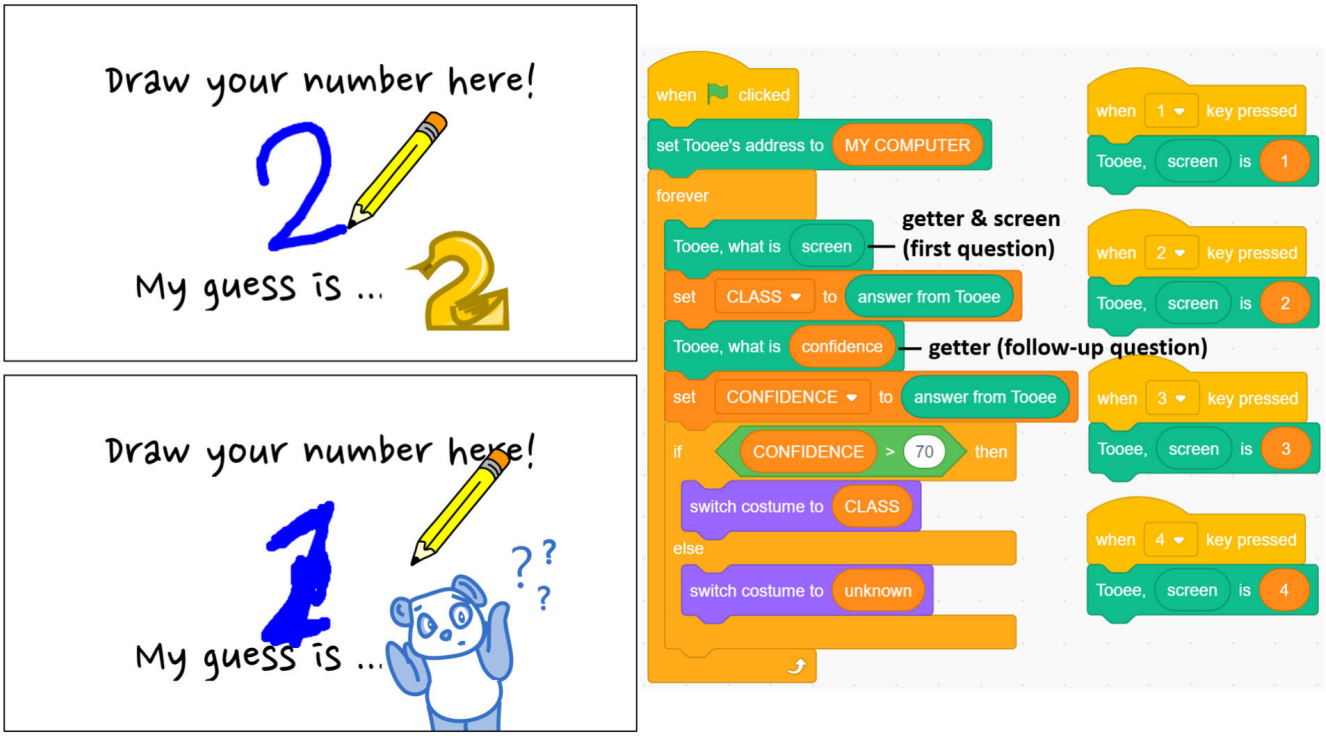


FIGURE 10. Handwriting recognition example for K-12 AI education. When the program starts, the current screen is repeatedly sent to Tooee. If Tooee is confident that the screen represents a number, the Scratch program displays the corresponding number on the screen. Otherwise, it displays an image indicating that the handwriting was not recognized as a number. The user can train Tooee by pressing the "1," "2," "3" or "4" key.

TABLE 2. The code complexity score (calculated based on *Dr. Scratch*) for each example application. The terms "ABS," "PAR," "LOG," "SYN," "FLO," "USE," and "DAT" are abbreviations for *abstraction and problem decomposition, parallelism, logical thinking, synchronization, flow control, user interactivity,* and *data representation,* respectively.

Area	Program	ABS	PAR	LOG	SYN	FLO	USE	DAT	Total
K-12 Big Data	Weather Forecast	1	3	0	3	2	2	2	**13**
	Top 5 Movies in Theaters	1	3	0	3	2	2	2	**13**
	COVID-19 Dashboard	1	0	3	2	2	2	2	**12**
	Saving Quiz Results to a CSV File	1	3	3	2	2	2	3	**16**
K-12 AI	Facial Image Classification	2	1	3	2	2	3	2	**15**
	Color Classification	0	0	0	2	2	3	2	**9**
	Object Classification	3	3	3	3	2	2	2	**18**
	Handwriting Recognition	1	1	3	2	2	2	2	**13**

B. EXPERIMENTAL RESULTS

Figure 12 summarizes the survey responses of the teachers to the pre/post questions shown in Table 3. According to the survey results, Tooee scored higher on all five evaluation measures (code complexity, usability, performance, extensibility, and deployability) compared to the existing environments for K-12 big data and artificial intelligence education. In the following subsections, we describe and analyze in detail the results of each evaluation measure.

1) CODE COMPLEXITY

The mean and standard deviation of the code complexity scores for the existing environments are 3.1 and 0.99, respectively, whereas the corresponding outcomes for Tooee are 4.0 and 0.76. The difference is statistically significant based on a two-tailed paired t-test with an alpha level of 0.05 ($p = 0.01746$). Here, all but two teachers gave Tooee a score of 4 or 5, expecting K-12 students easily to be able to implement big data and artificial intelligence programs using Tooee.

Many teachers mentioned that block interfaces are appropriate for K-12 education. (1) Five teachers reported that the block interfaces were intuitive. (2) One teacher stated that the minimum number of additional blocks would be effective for K-12 education, and (3) another teacher noted that the example characters were cute and that students would have fun making example applications. However, some teachers stated that students should acquire background knowledge before using Tooee: five teachers reported that K-12 students had to study Scratch in advance to make these example applications,

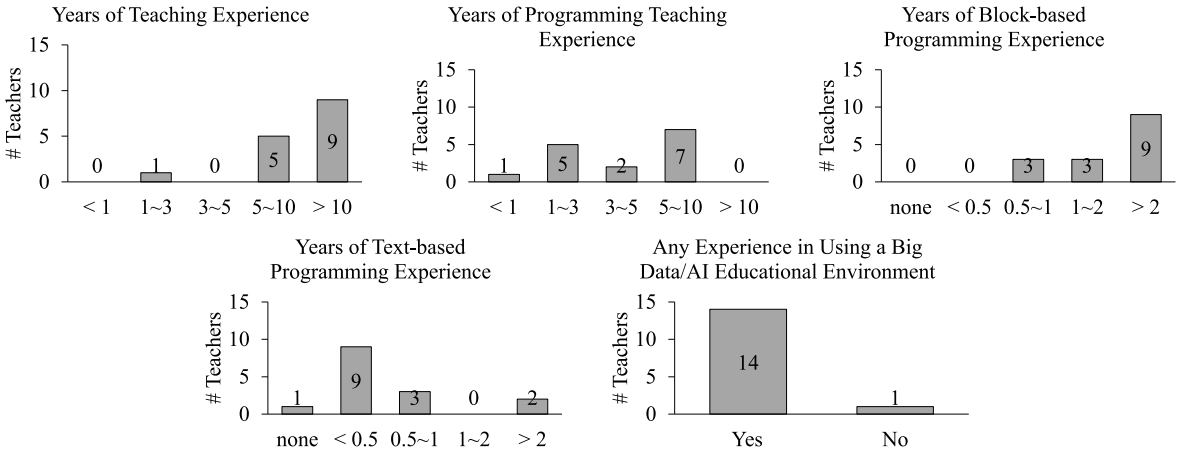


FIGURE 11. Statistics on the teaching and programming experience of the 15 teachers who responded the survey questions. Most of the teachers had at least five years of teaching experience and at least one year of programming teaching experience. Also, most of the teachers were familiar with block-based programming languages and had experience with text-based programming languages and big data/artificial intelligence educational environments.

TABLE 3. Survey questions for the five evaluation measures given to the 15 teachers in this study. Before using Tooee, the teachers evaluated existing educational environments on five evaluation measures. After using Tooee, the teachers evaluated Tooee on the same evaluation measures.

Measure	*[Pre or Post]* Questions
Code Complexity	Do you think K-12 students can easily implement big data/artificial intelligence programs using *[existing environments or Tooee]*?
Usability	Do you think K-12 students can conveniently use *[existing environments or Tooee]*?
Performance	Do you think the execution speed of *[existing environments or Tooee]* is fast enough for K-12 big data and artificial intelligence education?
Extensibility	Do you think it will help K-12 students create more diverse programs when they use *[existing environments or Tooee]*?
Deployability	Do you expect K-12 students easily to be able to share their own big data or artificial intelligence programs created using *[existing environments or Tooee]* with others?

and one teacher held that the exercise would be appropriate for students in the fourth grade or higher who already understand the concepts ''files'' and ''folders.'' On the other hand, there was also the opinion that given the intuitiveness of the block interface, no background knowledge was required for students.

2) USABILITY

While the mean and standard deviation for the Usability scores for the existing environments are 3.0 and 1.07, respectively, the corresponding scores for Tooee are 3.9 and 0.83. The difference is statistically significant based on a two-tailed paired t-test with an alpha level of 0.05 ($p = 0.01746$). Tooee was awarded 2, 3, 4, and 5 points by one, three, eight, and three teachers, respectively. In addition, many teachers noted that it would not be difficult for students to access *http://tooee.org* and that because Tooee's menu structure was highly similar to that of Scratch, they could easily adapt to the new menu. However, there was also the opinion that a teacher's guidance is occasionally necessary because young students are not as accustomed to using desktop computers compared to how well they use mobile devices.

3) PERFORMANCE

In terms of performance (execution speed), the average score for the existing environments and that of Tooee showed a very large difference (p-value: 0.00526). However, this is not a fair comparison because it does not take into account the accuracy of the machine learning algorithms provided by default.

All teachers except one gave a score of 4 or 5, suggesting that sending and receiving large amounts of data through WebSocket communications can be effective for the purpose of K-12 education. While one teacher only gave Tooee a score of 2 due to its slow training speed, two teachers noted that the amount of time it takes to train a machine learning model was acceptable for K-12 education. One teacher also stated that Tooee worked quickly even on a computer installed in his primary school.

4) EXTENSIBILITY

The mean and the standard deviation for the extensibility scores in the existing environments are 4.0 and 1.20, respectively, whereas the corresponding scores for Tooee are 4.4 and 0.99. Although the difference is not statistically significant based on a two-tailed paired t-test with an alpha level of 0.05 ($p = 0.18872$), the mean score of Tooee is 0.4 points higher than that for the existing environments.

Teachers who gave high scores on the extensibility measure for Tooee stated that K-12 students could likely create diverse programs very easily using Tooee because they could use interesting functionalities not previously provided in Scratch simply by writing a small amount of code. Some of the teachers also stated that they would expect students to

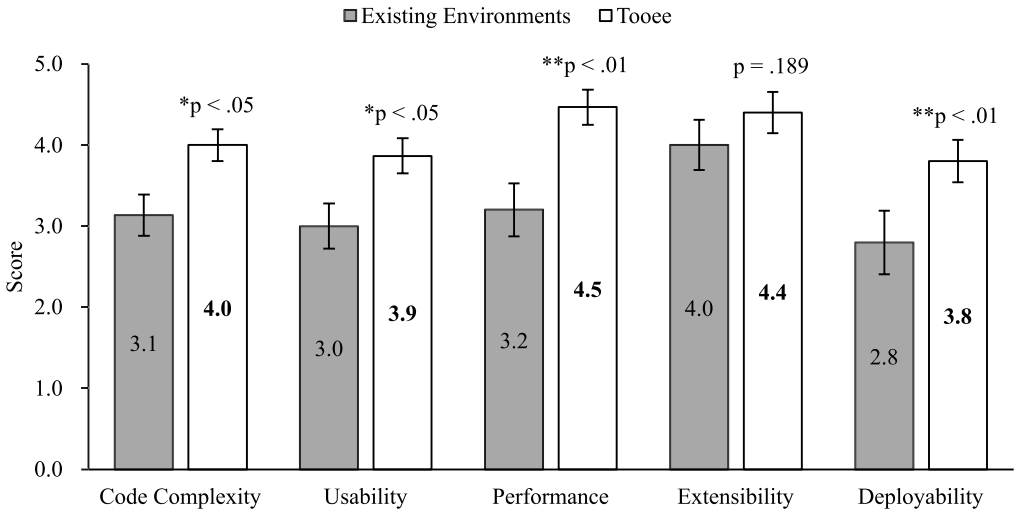


FIGURE 12. Experimental results comparing existing environments and Tooee for five evaluation measures. The scores indicate the mean scores of the teachers' responses. Tooee scored higher for all five evaluation measures, and significant differences were observed between the scores of existing environments and Tooee on four evaluation measures ($p < 0.05$).

be able to create more practical programs that are closely related to our everyday lives as they can deal with a variety of data using Tooee. One teacher noted that he expects Tooee to be particularly helpful for students who are familiar with programming.

On the other hand, teachers who assigned low scores on the extensibility measure for Tooee stated that it would not be easy for students to write Python (or another text-based programming language) code in order to add new functionalities to Tooee. However, if teachers or programmers create and distribute Tooee with added functionalities, students would not necessarily need to write Python code on their own.

5) DEPLOYABILITY

For the deployability score of the existing environments, the teachers assigned a score of 2.8 (SD: 1.52) while for Tooee the corresponding value was 3.8 (SD: 1.01). The difference is statistically significant based on a two-tailed paired t-test with an alpha level of 0.01 ($p = 0.00593$). Ten out of 15 teachers positively rated Tooee's deployability (4 or 5 points). They expected that students who had experience with Scratch programs on the *https://scratch.mit.edu* website would easily be able to distribute the program made by Tooee. On the other hand, teachers who did not give Tooee a high deployability score responded that it may be difficult for young students to understand the process of creating an HTML file because they may not be familiar with the concepts of files and folders.

VII. CONCLUSION

In this paper, we proposed a novel Scratch extension named *Tooee* that allows Scratch to communicate with text-based programming languages. Based on four design principles, the extension is composed of a minimum number of conversational blocks. Our extension greatly enhances the extensibility of Scratch so that K-12 students can create various types of big data or AI-related applications. We presented eight example applications that young students can easily implement. We also analyzed the strengths and weaknesses of this extension. The experimental results show that our approach is effective in K-12 big data and artificial intelligence education.

We can summarize our main contributions of this paper by answering the research questions: (1) We demonstrated that we can create a variety of big data/artificial intelligence programs using a block-based programming environment. When using the Tooee extension, because we can utilize the functionalities provided by text-based programming languages, it is now possible to implement a variety of programs even with a block-based programming language. (2) Also, We analyzed the advantages of our Tooee extension compared to other big data/artificial intelligence educational environments through the analyses and teacher surveys.

A limitation of this study is that it did not provide various types of teaching materials such as teacher guides, lesson plans, and rubrics. Therefore, in addition to the eight example applications and a ten minute video tutorial, we plan to create and distribute more diverse resources that teachers can use in their own classrooms. The Tooee extension and its related materials will be distributed on the Tooee website (*http://tooee.org*).

REFERENCES

[1] D. Lane, *Machine Learning for Kids: An Interactive Introduction to Artificial Intelligence*. San Francisco, CA, USA: No Starch Press, 2021.

[2] S. Druga, "Growing up with AI: Cognimates: From coding to teaching machines," Ph.D. dissertation, Dept. Program Media Arts Sci., Massachusetts Inst. Technol., Cambridge, MA, USA, 2018. [Online]. Available: https://dspace.mit.edu/handle/1721.1/120691

[3] M. Carney, B. Webster, I. Alvarado, K. Phillips, N. Howell, J. Griffith, J. Jongejan, A. Pitaru, and A. Chen, "Teachable machine: Approachable web-based tool for exploring machine learning classification," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, Apr. 2020, pp. 1–8.

[4] J. Lee and S. Lee, "A study on experts' perception survey on elementary AI education platform," *J. Korean Assoc. Inf. Educ.*, vol. 24, no. 5, pp. 483–494, Oct. 2020.
[5] N. Alturayeif, N. Alturaief, and Z. Alhathloul, "DeepScratch: Scratch programming language extension for deep learning education," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 642–650, 2020.
[6] K. Kahn, R. Megasari, E. Piantari, and E. Junaeti, "AI programming by children using snap! Block programming in a developing country," in *Proc. 13th Eur. Conf. Technol. Enhanced Learn.* Leeds, U.K., 2018, pp. 1–14.
[7] R. Williams, H. W. Park, L. Oh, and C. Breazeal, "PopBots: Designing an artificial intelligence curriculum for early childhood education," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 9729–9736.
[8] J. D. Rodríguez García, J. Moreno-León, M. Román-González, and G. Robles, "LearningML: A tool to foster computational thinking skills through practical artificial intelligence projects," *Revista de Educación a Distancia (RED)*, vol. 20, no. 63, pp. 1–37, Apr. 2020.
[9] A. Agassi, H. Erel, I. Y. Wald, and O. Zuckerman, "Scratch nodes ML: A playful system for children to create gesture recognition classifiers," in *Proc. Extended Abstr. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–6.
[10] A. Rao, A. Bihani, and M. Nair, "Milo: A visual programming environment for data science education," in *Proc. IEEE Symp. Vis. Lang. Human-Centric Comput. (VL/HCC)*, Oct. 2018, pp. 211–215.
[11] (2021). *Big Data and Artificial Intelligence Educational Materials Using Entry*. [Online]. Available: https://blog.naver.com/entrylabs
[12] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
[13] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, "The Scratch programming language and environment," *ACM Trans. Comput. Educ.*, vol. 10, no. 4, pp. 1–15, Nov. 2010.
[14] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, pp. 33–35, 2006.
[15] (2017). *ACM CSTA K-12 Computer Science Standards*. [Online]. Available: https://portal.ct.gov/-/media/SDE/CTE/CSTA-K12-Computer-Science-Standards-Revised-2017.pdf
[16] Y. Park and Y. Shin, "Comparing the effectiveness of scratch and app inventor with regard to learning computational thinking concepts," *Electronics*, vol. 8, no. 11, pp. 1269–1280, Nov. 2019.
[17] (2021). *Artificial Intelligence With MIT App Inventor*. [Online]. Available: https://appinventor.mit.edu/explore/ai-with-mit-app-inventor
[18] LEGO Education. *WeDo 2.0 Programming Block Descriptions*. Accessed: Jul. 1, 2021. [Online]. Available: https://education.lego.com/product-resources/wedo-2/downloads/programming-block-descriptions
[19] D. Wolber, H. Abelson, E. Spertus and L. Looney, *App Inventor*. Sebastopol, CA, USA: O'Reilly Media, 2011.
[20] D. Wolber, "App inventor and real-world motivation," in *Proc. 42nd ACM Tech. Symp. Comput. Sci. Educ. (SIGCSE)*, 2011, pp. 601–606.
[21] M. Tsur and N. Rusk, "Scratch microworlds: Designing project-based introductions to coding," in *Proc. 49th ACM Tech. Symp. Comput. Sci. Educ.*, Feb. 2018, pp. 894–899.
[22] J. Moreno-León and G. Robles, "Dr. Scratch: A web tool to automatically evaluate scratch projects," in *Proc. Workshop Primary Secondary Comput. Educ.*, Nov. 2015, pp. 132–133.
[23] J. Moreno-León, G. Robles, and M. Román-González, "Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking," *RED. Rev. Educ. Distancia*, vol. 46, pp. 1–23, Jan. 2015.
[24] J. Moreno-León and G. Robles, "Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills," in *Proc. Scratch Conf.* Amsterdam, The Netherlands, 2015, pp. 12–15.

YOUNGKI PARK received the B.S. degree in computer science from the Korea Advanced Institute of Science and Technology, in 2008, and the master's and Ph.D. degrees in computer science and engineering from Seoul National University, in 2010 and 2015, respectively. From 2015 to 2016, he was a Research Staff Member with the Samsung Advanced Institute of Technology. Since 2016, he has been an Assistant Professor and an Associate Professor with the Department of Computer Education, Chuncheon National University of Education. His research interests include computer science education, and K-12 big data and artificial intelligence education.

YOUHYUN SHIN received the B.S. degree in computer science education from Korea University, in 2013, and the Ph.D. degree in computer science and engineering from Seoul National University, in 2019. Since 2020, she has been an Assistant Professor with the Department of Computer Science and Engineering, Incheon National University. Her research interests include natural language processing, spoken language understanding, and technology driven learning.