mathematics

MDPI

Article

Development of CNN-Based Data Crawler to Support Learning Block Programming

HuiJae Park [1], JaMee Kim [2] and WonGyu Lee [3,*]

[1] Department of Computer Science and Engineering, Graduate School, Korea University, Seoul 136701, Korea; x2xjoe@gmail.com
[2] Major of Computer Science Education, Graduate School of Education, Korea University, Seoul 136701, Korea; celine@korea.ac.kr
[3] Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 136701, Korea
* Correspondence: lee@inc.korea.ac.kr; Tel.: +82-2-3290-2391

Abstract: Along with the importance of digital literacy, the need for SW(Software) education is steadily emerging. Programming education in public education targets a variety of learners from elementary school to high school. This study was conducted for the purpose of judging the proficiency of low school-age learners in programming education. To achieve the goal, a tool to collect data on the entire programming learning process was developed, and a machine learning model was implemented to judge the proficiency of learners based on the collected data. As a result of determining the proficiency of 20 learners, the model developed through this study showed an average accuracy of approximately 75%. Through the development of programming-related data collection tools and programming proficiency judging models for low school-age learners, this study is meaningful in that it presents basic data for providing learner-tailored feedback.

Keywords: computer education; programming education; leaner classification; log collection; Scratch

MSC: 68T10

Citation: Park, H.; Kim, J.; Lee, W. Development of CNN-Based Data Crawler to Support Learning Block Programming. *Mathematics* **2022**, *10*, 2223. https://doi.org/10.3390/math10132223

Academic Editors: Ioannis G. Tsoulos and Christophe Guyeux

Received: 30 April 2022
Accepted: 15 June 2022
Published: 25 June 2022

1. Introduction

According to a digital revolution in modern society, many countries have shown changes in education policy, such as starting SW education for elementary school students [1]. Starting with the UK in 2014, Korea in 2015, and Japan in 2017, many countries have included programming education in their elementary school curriculum [2–7]. Moreover, in OECD Education 2030, digital literacy along with literacy and numeracy are included as competencies for living in the current age in the same context [8]. To foster digital literacy, programming education for learners with a low learning age is conducted with block-based programming education rather than text-based programming [9–11].

The difficulties of programming education for novice learners have been confirmed through many studies [11–13]. The difficulty of the programming process is that when an error occurs, the cause is not known, so even if the problem is solved, the procedure is not correctly recognized. Without the help of the instructor, the cause of the error cannot be identified and easily missed. Scratch, an educational programming language, works through 'trial and error', and the learner's difficulty appears relatively low [14,15]. 'Trial and error' in Scratch includes the process of learning while debugging through sprites in real time. However, even if one has completed the program, if one is not aware of the process through which it was completed, the learning effect will be reduced. In other words, if a meta-cognition of what learners know and do not know on their own is formed in the programming process, it will contribute to enhancing the learning effect [16].

If errors that occur during programming can be checked and fed back, it will contribute to enhancing the learning effect. For learners, it is possible to keep their interest in programming by minimizing the decline in learning motivation. However, in the current programming learning situation, there is a limit to checking the programming process of all students because the instructor may be in charge of more than 20 students [17]. To support programming for adults and college students, peer evaluation and online judgement are being used. However, the support for the programming process is insufficient compared to the support for the results [18].

Studies have been conducted to analyze the learner's programming process and to support programming. It was analyzed based on the record of the learner's programming process. In other words, the programming process was analyzed by checking the final result of the code written by the learner or by recording and reviewing the learner's programming process [19]. In this case, rather than supporting the programming process in real time, there is a limit to support after the results are displayed.

In order to support programming learning, it is necessary to analyze the learner's programming process and provide real-time feedback to maintain the interest in learning [20]. Real-time support of programming learning should accompany the entire process of learners' programming and include various data analyses [21]. That is, if patterns of errors are identified in the programming process and feedback is provided at difficult points, the process will be able to support programming learning.

Therefore, this study conducted basic research to support learners in real time in the programming process [20]. In other words, it analyzed the learner's programming log to confirm points of difficulty in the programming process. The point of difficulty is used to classify the behavior types for programming and to provide appropriate feedback for each behavior type.

The CNN(Convolutional Neural Network)-based learning model checks whether it is possible to classify the programming log of a specific project activity by dividing it into a programming dataset of a proficient user and a programming dataset of an inexperienced user. Subsequently, the accuracy was extracted using the datasets of other students.

2. Related Work

Clues for determining whether learners are learning programming education correctly are often lacking, depending on the perspective. In a one-to-many setting, such as the teacher–student relationship in a school, various cues are missed or not collected. The data of programming learners from existing studies are used to verify whether a teacher is suitable for judging learners.

Evaluation and analysis studies to check whether the learner's programming process data affect the learner's learning level have been continuously conducted. In this section, we discuss the programming data analysis that this study intends to proceed with and related research on how to use the data.

2.1. Learning Process Analysis Study

There are many ways to analyze the programming process. S. Y. Lye and J. H. Koh conducted process analysis while the teacher directly watched the student's programming process or reviewed the video recording to assist the student's learning. As the number of hours of training increases and the scope of education increases, the scope of feedback required for students increases and the number of parts that learners do not know increases, so this method is inefficient and incorrect [19]. Moreno et al. proceeded to evaluate computational thinking ability based on the project file that results from the programming activity. This evaluation method based on output may be suitable for a test that evaluates only the result, but it does not sufficiently reflect the learner's programming intention or situation [22].

In one-to-many cases, systems that enable teachers to understand learners' programming processes must have educationally grounded skills. In order to confirm the hypothesis

that the programming process is related to the learner's achievement, this study confirmed a related study that left the programming process as data. J. Helminen and two others collected data of students solving Parsons programming problems. The web programming tool used in this study has a 'Get Feedback' button, which allows the learner to receive the feedback requested by clicking this button [23]. The researchers collected these records to gather data about situations in which the learners struggled. This method is effective for capturing the moment when feedback is needed, but data for estimation is insufficient because it does not collect all the processes up to that point. The study of D. Filva et al. traces the mouse trajectory that learners program in the Scratch editor [24]. The tracked path and click position data are collected as a single dataset, and the learner is classified as a designer type or a coder type. Due to the characteristics of the Scratch programming editor (the location of the programming block where changes have already been written), there is a limitation in that it is insufficient to distinguish learners. In the study of M. Kong et al. the behavioral data of learners programmed in Scratch were collected as one dataset on the before–after relationship [25]. Using the data collected in this way, we created data to predict what the next action will be when the learner takes a certain action. This study speculates on programming behavior that applies to anyone. Although this study has a good premise to generalize, it lacks clues about how learners exhibit the resulting behaviors in various situations. In this study, the entire programming process is used as continuous dataset to reflect both the before and after of the behavioral process.

2.2. User Data Classification Study

If a neural network model is created based on the user's pattern, the behavioral pattern of an invalidated user can be distinguished. When user behavior data are generated as two-dimensional data, the model shows high accuracy when using a convolutional neural network compared to other neural network models. Looking at the CNN proposed by Y. LeCun's research, the algorithm using a two-dimensional array does not require direct feature extraction and can be sufficiently processed as long as the computing power is improved, even when the data are increased [26].

In the study of G. Cao et al., a system that can accommodate sensor values of various channels on the face was built for facial expression recognition [27]. It was effective in recognizing facial expressions when using CNN by generating data obtained through the system as two-dimensional array data. The fact that each sensor datum, which cannot be regarded as irrelevant, can be distinguished even if it is displayed in two dimensions rather than a three-dimensional or more multidimensional array is closely related to this study.

Various types of CNN are available depending on the type of data input or the desired result. According to the research of F. Ning et al., 2D-CNN is the best to create a model for classifying 3D modeled mechanical parts. The result shows that the efficiency of the model created through preprocessing and trained through the CSV file is higher than the result obtained by learning the 3D file itself on the 3D-CNN model.

3. Development Methods

In this section, we set up the programming environment and analyze the form of data to be collected in order to check how to use the learner's programming log to analyze the process. A learning model was set up for the purpose of classifying these data and the skill level of the learner.

3.1. Development Environment

This study examines the relationship between the learner's programming process and the learner's proficiency. For research, we need data quantifying the learner's programming process and a learning model to analyze these data. A separate log collection tool is required for data collection. The tool used for log collection is a programming log crawler developed for this work. The programming log crawler processes data in two servers, as shown in Figure 1. The result of programming in the Scratch editor is transmitted as a JSON-type

script, and the JSON file is transmitted and saved from the client to the database server. If a learner inputs an index before programming, the learner and the project are stored separately in the database.

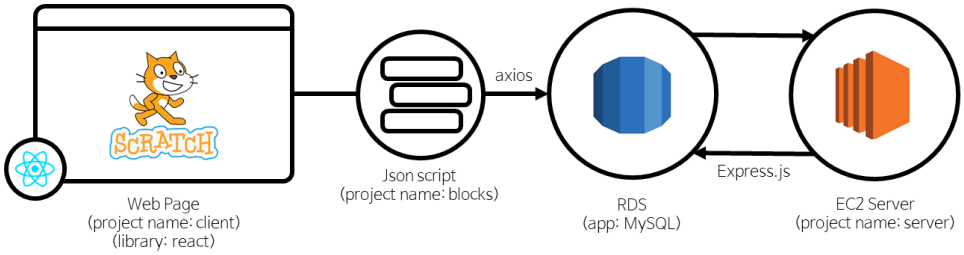

Figure 1. Data flow diagram of a programming log crawler.

Scratch was selected for this study because it is the most used language among block programming languages. The reason for choosing a block programming language is that, first, it is used as a learning language for novice learners, as it is burdensome to provide CUI (Character User Interface) in learning [28]. Programming learners who are accustomed to programming can recognize what they do not know, but novice learners cannot. A block programming language that is easy to visually understand what is not clear was selected. The essence of this study is to develop a tool that enables teachers to check the programming process and provide feedback to assist learners in learning. Second, the block programming language is effective in confirming the change that occurred at the time of the click. In a block programming language, it is difficult to perform more than two actions at a time when a single click occurs. By comparing the programming log created before the click with the programming log immediately after the click occurred, one can categorize what kind of action was taken. Using Scratch, one can easily represent the learner's programming process numerically compared to other string languages.

As a result of implementing the functions, it was possible to collect the learner's log through the editor, as shown in Figure 2.

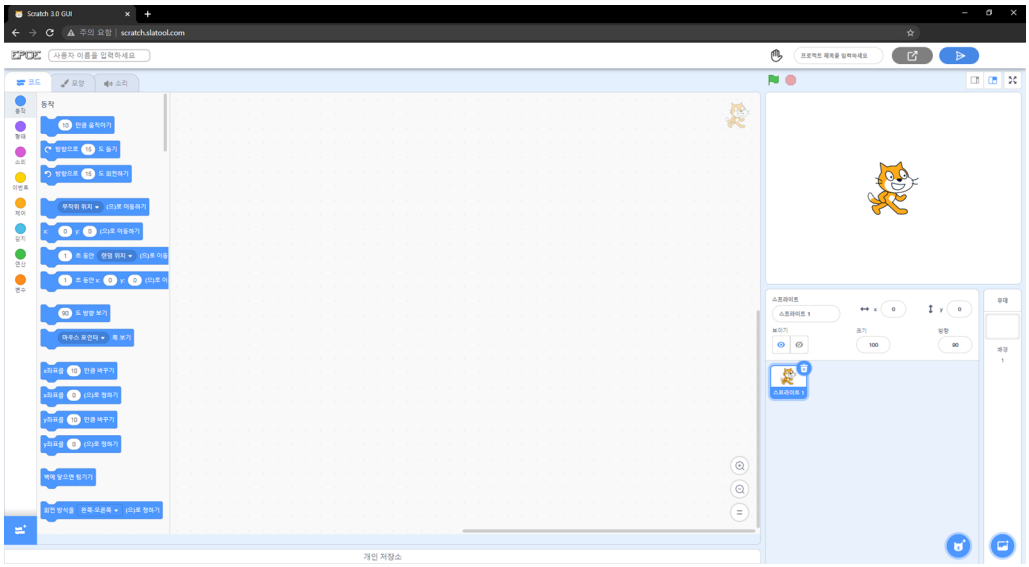

Figure 2. A view of the programming log crawler used in the study.

The raw data collected through the programming log crawler are difficult to distinguish with the naked eye. The programming log crawler analyzes the difference between the JSON script remaining before the learner clicks and the JSON script right after the click is made to understand the learner's behavior type. Data are extracted in CSV format by

classifying each learner from the database. When the extracted raw data are opened, they are composed as follows.

idx,project_json,created
21664, "{"meta": {"vm": . . . , "2021-06-16 17:22:36"}"
21666, "{"meta": {"vm": . . . , "2021-06-16 17:22:39"}"
. . .

In the data above, the log the learner programmed is in "{"meta":.... By parsing the programming log in this area and comparing the necessary data, the behavior type can be extracted. The extracted behavior types were classified into eight types of blocks including the first seven types: 'add code', 'delete code', 'code copy', 'code order change', 'code position change', 'code internal value change', and 'no change' and by adding an eighth type 'detection of other changes' as an extra. The data to be actually generated are composed of seven columns using seven types, excluding 'detection of extraneous changes'. The configured database is displayed as in Figure 3.

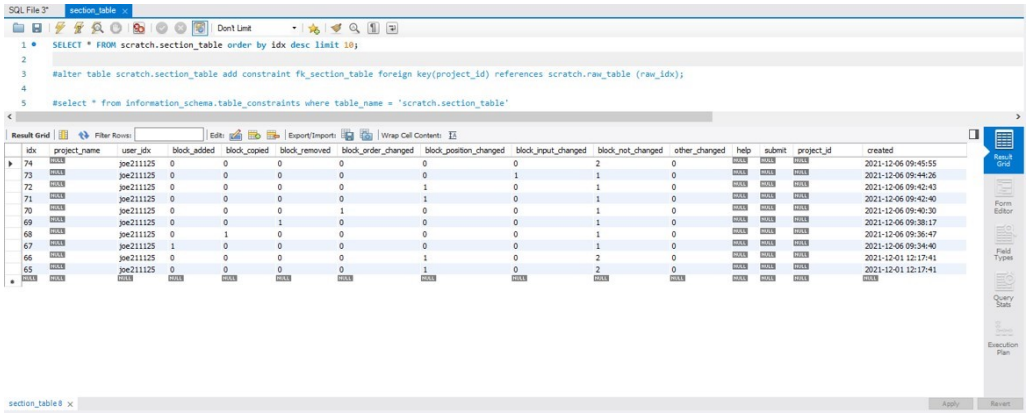

Figure 3. User log collection and database browsing.

Behavior type data consist of seven columns, and, because it creates as many rows as the number of mouse clicks that occurred during the programming process, the number of rows is different for each learner with high probability.

3.2. Learning Model

The Conv2d module based on the CNN model was used as a model for learning the data stored in the csv format of the learner's programming process. The Conv2d module has a key value that calculates the size of the output channel according to the size and factor of the input channel. The Conv2d module uses the convolution layer, which is typically used in image classification, which has a key value that calculates the size of the output channel according to the size and factor of the input channel [29].

Using the Conv2d module, we obtained the kernel corresponding to the programming log of the 'mature' learner and the programming log of the 'immature' learner. The kernel, which corresponds to the two groups, makes it possible to distinguish whether a user is an experienced user or an inexperienced user when new learning log data are received.

The difference from a commonly used model in creating a learning model is that the training data are not of the same size. Depending on the learner, even if the same result occurs, the number of clicks during the programming process may vary. Unlike the process of removing or resizing unnecessary parts such as image pre-processing, there are no data that can be omitted in the programming process. Therefore, it was necessary to recognize the maximum data size in advance and fill the remaining data with empty tokens.

4. Implementation of Development

In this section, we tried to confirm the usefulness of the model by securing data and learning through the developed tool. After recruiting actual learners, securing the learner's

metadata, and guiding the project to be carried out, project activity data of 20 people were collected. Next, in order to learn by putting the generated data into the learning model, the model was customized.

4.1. Data Collection and Further Processing

The result we wanted to achieve through the artificial intelligence neural network model was to check whether the data of the programming process can be classified according to the learner's programming proficiency. In order to classify the proficiency of the learner, it is difficult to proceed with the data programmed by the learner without a specific purpose. This is because it is difficult to identify behavioral patterns or commonalities appearing in the programming process.

The creation of a specific task is required to generate data to feed into the learning model. The students participating in this task are third and fourth graders attending elementary school A in Seoul. In order to proceed with this task, students conducted an eighth session of using the Scratch function. Below is the programming task used to collect the data.

- Cat sprite is centered on the left and can move in all directions;
- Apple sprite is pinned to the top center;
- If a user clicks the flag ( ), only cats and apples are visible;
- User moves to the next background when touching the right edge of the background;
- There is a star sprite in the center right of the moved background;
- Task ends when the cat touches the star sprite.

Table 1 shows the grade, gender, and programming education experience of the students who performed the task.

Table 1. Students participating in the experiment and their experience in programming education.

No.	Student	Grade	Sex	Programming Experience
1	A	3	M	-
2	B	3	F	O
3	C	3	M	O
4	D	3	M	X
5	E	3	F	-
6	F	3	M	O
7	G	3	M	O
8	H	3	M	O
9	I	3	F	O
10	J	3	M	O
11	K	3	M	-
12	L	3	M	O
13	M	4	M	X
14	N	4	F	O
15	O	4	M	X
16	P	4	M	O
17	Q	4	F	O
18	R	4	M	O
19	S	4	M	O
20	T	4	F	O

-: Programming for the first time; O: Scratch for the first time; X: Scratch experienced.

Table 1 is the evaluation of students' project results. Considering the basic functions and difficult parts to implement in the course of performing the task, we were able to qualitatively evaluate the student's proficiency.

Most of the students implemented the movement of moving the sprite to the left or right well, but the function to make it disappear when the sprites touched or to appear when the project was executed again was not well implemented. The standards of proficient

and inexperienced students were reflected in the standards by checking whether they work as well as in the beginning when the project is restarted.

According to Table 2, students selected as proficient users are subjects D, E, H, J, L, M, O, Q, R, and S, and students selected as inexperienced users are subjects A, B, C, F , G, I, K, N, P, and T.

Table 2. Results of project activities of students who participated in the experiment.

No.	Stdt	Move-Left	Move-Right	Jump	Interaction	Disappear	Can Restart	Configure	Design	Finish on Time
1	A					Failed to Submit				
2	B	O	O	O	O	O	X	O	X	O
3	C	O	O	O	O	O	-	O	O	O
4	D	O	O	O	O	O	O	O	O	O
5	E	O	O	O	O	O	O	O	-	O
6	F	-	X	O	O	O	X	X	O	X
7	G	O	O	O	O	O	X	X	O	O
8	H	O	O	O	O	O	O	O	O	O
9	I	O	O	X	O	O	X	O	X	O
10	J	O	O	O	O	O	O	O	X	O
11	K	X	X	O	O	O	X	X	O	X
12	L	O	O	O	O	O	-	O	O	O
13	M	O	O	O	O	O	O	O	X	O
14	N	O	O	X	O	O	-	O	O	X
15	O	O	O	O	O	O	O	O	O	O
16	P	O	O	X	O	O	X	O	O	O
17	Q	O	O	O	O	O	O	O	X	O
18	R	O	O	O	O	O	O	O	O	O
19	S	O	O	O	O	O	O	O	O	O
20	T	-	O	X	X	O	X	O	X	O

-: Implemented but not working. O: Succeed result. X: Failed result.

The process of creating a file by importing students' data is shown in Figure 4.

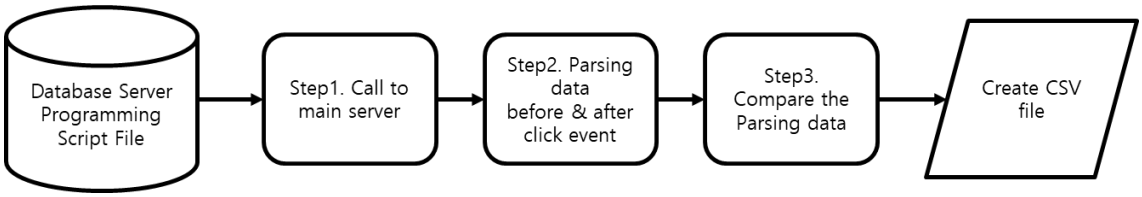

Figure 4. The whole process of collecting data and creating a CSV file.

The learner's programming log stored in the database server is in the form of a JSON script. These data need to be called to the main server, but because a comparison target is needed, all data before and after the click are loaded. In the loaded JSON script, only the necessary data are parsed and then the data are compared. This compared result value is created and saved in CSV format. The CSV file consists of the same seven columns as in Figure 5, but it is composed of different rows because the number of clicks used for the task is different for each student.

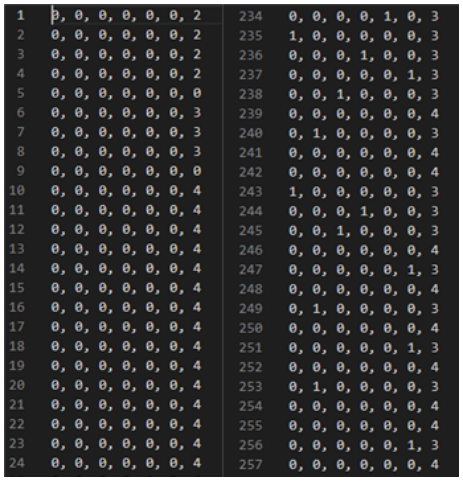

Figure 5. The actual composition of the CSV file.

A few errors occurred in the process of creating the CSV file. Because these errors were not a problem with the parser, but a problem with the development environment or the generated data itself, exception handling was necessary. First, Python's JSON library sometimes caused errors in the encoding process. If there are consecutive quotation marks (“,”) used to recognize output statements or commas (,) used to separate data in JSON files, they are sometimes removed while encoding. An exception was handled for this, and in this case, the corresponding row was skipped.

Some of the generated CSV files were used as training data and some were used as validation data. In the neural network model, the tree of the folder was aligned so that the data used for training and the data used for validation could be distinguished.

4.2. Learning Model

After creating the dataset, the data must be placed into the neural network to train it. The dataset is transferred to the neural network through the data loader. The CNN model must create a layer according to the shape of the data and determine the activation function.

The learning model used in this study is a 2D-CNN model that learns through two-dimensional data. The data to be classified through learning does not use a linear model. Because the result value to be distinguished is a binary identifier that distinguishes whether the corresponding data are 0s or 1s, a sigmoid function that outputs a value between 0 and 1 was used. In addition, ReLU was used together so that a value less than 0 could not occur.

The configuration of the model using the activation function value is as follows.

```
self.layer1 = nn.Sequential(nn.Conv2d(1, 64, 3),
nn.ReLU(),
nn.Conv2d(64, 256, 3),
nn.ReLU(),
nn.Conv2d(256, 512, 3))
self.layer2 = nn.Sequential(nn.Linear(512,256),
nn.ReLU(),
nn.Linear(256,64),
nn.ReLU(),
nn.Linear(64, 1),
nn.Sigmoid())
```

The generated data are learned through the model. Through the training model, the training loss and validation loss are printed out every time the epoch runs.

5. Results

The loss output through the training model file is shown on the left graph in Figure 6. This training model noticeably reduced the loss up to about 40 epochs. The validation loss also decreased until the 40th epoch, but after that, the loss value showed a tendency to rise again. Relatively, compared to the training loss, the validation loss does not decrease, but rises.

The result of measuring the accuracy for each epoch through the generated model is as shown in the graph on the right of Figure 6. In the case of the training model in which the loss was converging to 0, the accuracy was 100% from 11 epochs. In the case of the validation model, after measuring 70% accuracy for the first 10 epochs, unstable results showed 40% accuracy at 20 epochs. After 50 epochs, the accuracy was 70–80%.

The reason that the region showing the 'highest accuracy' cannot be accurately presented is because there are not enough data to create a model, so the point with the highest accuracy of data appears redundantly. (Ten data points were used per classification target.) Therefore, this study conducted a single T-test to find out the average value of the high hit rate in a small number of samples and to check whether the value of the view has normality.

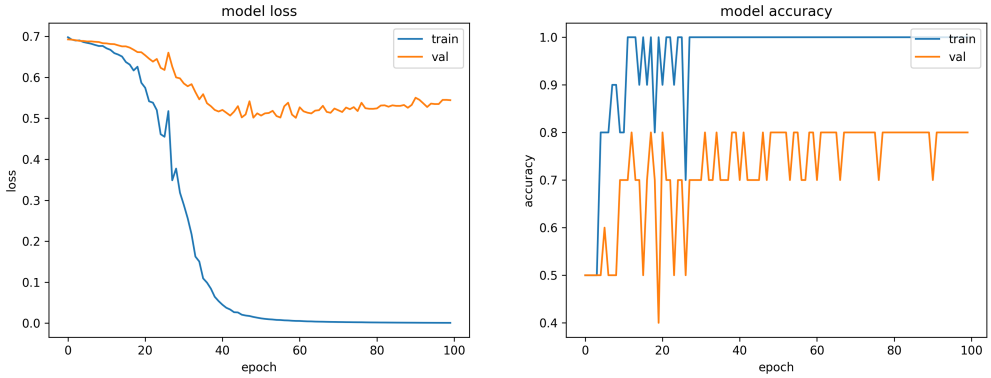

Figure 6. Loss and accuracy values of the model trained through the Conv2d model.

A single T-test was performed using the accuracy data corresponding to each epoch of validation accuracy and training accuracy. As a result of analyzing the existing data to present a test value that is close to the existing mean, the mean and standard deviation of validation accuracy and training accuracy are as shown in Table 3.

Table 3. Mean and standard deviation of training accuracy and validation accuracy.

	N	Average	Deviation
Training accuracy	100	0.959	0.111
Validation accuracy	100	0.728	0.100

The significance probability was confirmed by changing the values close to the two average values in units of 0.1. Table 4 is a single T-test result corresponding to training accuracy, and Table 5 is a single T-test result corresponding to validation accuracy.

Table 4. Significance probability according to test value of training accuracy.

Test Value	T-Value	Significance Probability
0.91	4.411	0.000
0.92	3.510	0.001
0.93	2.610	0.010
0.94	1.710	0.090
0.95	0.810	0.420
0.96	−0.090	0.928
0.97	−0.990	0.325
0.98	−1.890	0.062
0.99	−2.790	0.006
1	−3.690	0.000

Table 5. Significance probability according to test value of validation accuracy.

Test Value	T-Value	Significance Probability
0.69	3.778	0.000
0.7	2.784	0.006
0.71	1.790	0.077
0.72	0.795	0.428
0.73	−0.199	0.843
0.74	−1.193	0.236
0.75	−2.187	0.031
0.76	−3.181	0.002
0.77	−4.176	0.000

The highest value where the significance probability satisfies the normality and there is no statistically significant difference corresponds to the training accuracy of 96% and the validation accuracy of 73%.

6. Conclusions

Various educational programming languages have been developed so that learners without programming experience can have an interest in programming. Among the various languages developed, a block-based programming language such as Scratch has the advantage that it can be used by learners of low school age. In the case of a block-based programming language learned through trial and error, the learner directly recognizes the point of error, but it is difficult to help acquire basic programming-related knowledge.

Especially in the case of beginner learners, it is difficult to know where they are getting the error because they do not know what they do not know and where they made a mistake. The form used in the past—the form in which the learner gives simple feedback by moving a specific block or when an error occurs—can help to learn the function to solve the problem from a programming point of view, but it is difficult to acquire and understand the concept. Existing previous studies have also made efforts to support programming learning by analyzing the learner's error points or by analyzing the programming results. However, it was not possible to directly analyze the learner's programming process and provide error feedback according to the level of programming proficiency. In other words, the analysis of the programming process could not proceed.

This study tried to prepare basic data to judge the proficiency of learners based on the data on the programming process and to provide customized feedback according to the proficiency of the learners. Therefore, we developed a data collection tool for the learner's programming process and developed a model for judging the learner's proficiency by analyzing the collected data.

This study used an artificial intelligence neural network to analyze the log of the entire programming learning process. In the future, it is necessary to find a way to extract patterns for programming from each dataset through the learning model developed in this study and to provide appropriate feedback to learners according to the patterns. The significance of this study is that in public education, basic research was conducted to allow learners to engage in programming activities on their own in a state where teachers could not take care of all students. In addition, if factors such as the learner's personal background or learning propensity are collected, qualitative research can be expected to analyze the difference in which each factor affects accuracy.

Author Contributions: Conceptualization, H.P. and W.L.; methodology, H.P., J.K. and W.L.; software, H.P.; validation, W.L., J.K. and H.P.; writing—original draft preparation, H.P.; writing—review and editing, H.P. and J.K.; supervision, J.K. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A2C2013735).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Benešová, A.; Tupa, J. Requirements for education and qualification of people in Industry 4.0. *Procedia Manuf.* **2017**, *11*, 2195–2202. [CrossRef]
2. DfE. *National Curriculum in England: Framework for Key Stages 1 to 4, 2014*; UK Ministry of Education: London, UK, 2014.
3. DFE, U. National curriculum in England: computing programmes of study. *Retrieved July* **2013**, *16*, 2014.
4. Kim, J.; Lee, W. An analysis of the 2016 government guidelines for teaching of Japan and the outline of the 2015 revised curriculum of Korea. *J. Korean Assoc. Comput. Educ.* **2017**, *20*, 1–14.
5. *Summary of Deliberations So Far for the Next Course of Study (Report)*; Ministry of Education, Culture, Sports, Science and Technology: Tokyo, Japan, 2016; Volume 8, pp. 129–132.
6. High School Learning Guidance Guidelines; Ministry of Education, Culture, Sports, Science and Technology. Notification of 2018. 2017. Available online: https://www.mext.go.jp/en/about/pablication/__icsFiles/afieldfile/2019/03/13/1374478_001.pdf (accessed on 29 April 2022).
7. Practical Course (Technology and Home Economics)/Informatics Curriculum, Ministry of Education Notice No. 2015-74. 2015. Available online: www.koreaneducentreinuk.org/wp-content/uploads/2021/02/The-National-Curriculum-for-the-Primary-and-Secondary-Schools-2015.pdf (accessed on 29 April 2022).
8. OECD. *21st-Century Readers*; OECD: Paris, France, 2021 ; p. 213. [CrossRef]
9. Zhang, L.; Nouri, J. A systematic review of learning computational thinking through Scratch in K-9. *Comput. Educ.* **2019**, *141*, 103607. [CrossRef]
10. Nouri, J.; Zhang, L.; Mannila, L.; Norén, E. Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Educ. Inq.* **2020**, *11*, 1–17. [CrossRef]
11. Robins, A.; Rountree, J.; Rountree, N. Learning and teaching programming: A review and discussion. *Comput. Sci. Educ.* **2003**, *13*, 137–172. [CrossRef]
12. Wang, W.; Kwatra, A.; Skripchuk, J.; Gomes, N.; Milliken, A.; Martens, C.; Barnes, T.; Price, T. Novices' Learning Barriers When Using Code Examples in Open-Ended Programming. In Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1, Virtual Event, Paderborn, Germany, 26 June–1 July 2021; pp. 394–400.
13. Loksa, D.; Xie, B.; Kwik, H.; Ko, A.J. Investigating novices' in situ reflections on their programming process. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland, OR, USA, 11–14 March 2020; pp. 149–155.
14. Bhaugeerutty, V.S. Difficulties in Learning and Teaching Programming at Lower Secondary Level in Mauritius. *J. Contemp. Res. Soc. Sci.* **2021**, *3*, 48–61. [CrossRef]
15. Milne, I.; Rowe, G. Difficulties in learning and teaching programming—views of students and tutors. *Educ. Inf. Technol.* **2002**, *7*, 55–66. [CrossRef]
16. Davidson, J.E.; Deuser, R.; Sternberg, R.J. The role of metacognition in problem solving. In *Metacognition: Knowing About Knowing*; MIT Press: Cambridge, MA, USA, 1994, Volume 207, p. 226.
17. McDonald, C. Why is teaching programming difficult? In *Higher Education Computer Science*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 75–93.
18. Bagheri, A.; Hjorth, P. Planning for sustainable development: a paradigm shift towards a process-based approach. *Sustain. Dev.* **2007**, *15*, 83–96. [CrossRef]
19. Lye, S.Y.; Koh, J.H.L. Case studies of elementary children's engagement in computational thinking through scratch programming. In *Computational Thinking in the STEM Disciplines*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 227–251.
20. Long, J. Just For Fun: using programming games in software programming training and education. *J. Inf. Technol. Educ. Res.* **2007**, *6*, 279–290. [CrossRef]
21. Konecki, M. Problems in programming education and means of their improvement. In *DAAAM International Scientific Book*; DAAAM International: Vienna, Austria, 2014; Volume 2014, pp. 459–470.
22. Moreno-León, J.; Robles, G.; Román-González, M. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Rev. Educ. Distancia* **2015**, *46*, 1–23.
23. Helminen, J.; Ihantola, P.; Karavirta, V.; Malmi, L. How do students solve parsons programming problems? An analysis of interaction traces. In Proceedings of the ninth annual international conference on International computing education research, Auckland, New Zealand, 9–11 September 2012; pp. 119–126.
24. Filvà, D.A.; Forment, M.A.; García-Peñalvo, F.J.; Escudero, D.F.; Casañ, M.J. Clickstream for learning analytics to assess students' behavior with Scratch. *Future Gener. Comput. Syst.* **2019**, *93*, 673–686. [CrossRef]
25. Kong, M.; Pollock, L. Semi-Automatically Mining Students' Common Scratch Programming Behaviors. In Proceedings of the Koli Calling'20: 20th Koli Calling International Conference on Computing Education Research, Koli, Finland, 19–22 November 2020; pp. 1–7.
26. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]

27. Cao, G.; Ma, Y.; Meng, X.; Gao, Y.; Meng, M. Emotion recognition based on CNN. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 8627–8630.
28. Dillane, J. Frame-Based Novice Programming. In Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education, Trondheim, Norway, 15–19 June 2020; pp. 583–584.
29. Kannojia, S.P.; Jaiswal, G. Effects of varying resolution on performance of CNN based image classification: An experimental study. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 451–456. [CrossRef]