All Sciences Proceedings

1st International Conference on Pioneer and Innovative Studies

June 5-7, 2023 : Konya, Turkey

ICPIS

All Sciences Proceedings
http://as-proceeding.com/



https://as-proceeding.com/index.php/icpis

# Results of the research of algorithm creating capabilities using convolutional neural network

Anikó Apró

*Information Technology, Faculty of Informatics, University of Debrecen, Hungary*

*apro.aniko@inf.unideb.hu*

***Abstract***— The current technological momentum indicates that demand will soon rise for programming skills. For this reason, finding young individuals with these skills and orienting them towards programming related career paths will be highly important. In our research, we developed an age and programming knowledge independent system, which can help to identify talented youngsters. Before filling the test, we collected information through an anonymous survey about the test subjects' IT knowledge, which can help identify further correlations at the time of result evaluation. First we had to create a measurement system which will provide objective test results about the children's algorithmic thinking. We developed a web application that is responsive with a camera which follows the client eye moving and gather data into our database. This data is then passed into a machine learning algorithm to create a model and from it we could read our conclusions. The main goal of machine learning using convolutional neural network is predicting how the algorithmic thinking will develop in the case of children. We used a more game-like approach because it is more natural to children to think when they think they are playing some game. The game is a simple but effective LED lightning game and is very similar to a memory-based game.

*Keywords – Algorithm, Creation, Algorithmic Thinking, Measurement System, Skill, Ability, Talent*

## I. INTRODUCTION

Through the whole vertical spectrum of IT education, we can observe that the algorithmizing capabilities of students can vary greatly, even with

ds. This type of variation can also be observed amongst university students when they are learning programming. This phenomenon is not necessarily tied to the individual's learning affinity. Capabilities are largely influenced by skills, which are formed through genetic and early social processes alike.

It is also known that the most important component in forming programming capabilities is the development of algorithmizing skills. These skills can already be examined in the early childhood years, mostly determined by genetics, then they are further refined through other acquired skills as the individual gets older.

similar educational background improving the efficiency of programming education from basic to higher levels of studies.

During our earlier research projects, we developed a computer technology called light programming, which can help us to decouple the measurement results from the subjects' age and level of programming knowledge.

The better understanding of how algorithm creation capabilities are formed can lead us closer to

## II. THEORETICAL BACKGROUND

### A. *The need for early identification*

Capability is often confused with skill and experience. Capabilities are jointly formed by inherited traits as well as social factors. Contrary to that, skills are acquired through learning and

repeated practice,[1] which is a somewhat longer procedure, but once we have them – for example logical thinking -, they stay with us over the long term. There are capabilities which all of us have, but to a smaller or larger extent, such as intelligence and creativity. However, there are some which we call talents, these are capabilities that only certain people exhibit to a high standard. Examples include dexterity and musical hearing. To summarize, capabilities are things that cannot be taught, but can be developed. On the other hand, skills are things that we can acquire through learning and practice. Our acquired skills are present in the automatized part in all our actions. They make it possible to focus our attention on reaching the goal of our activity as well as on the road leading to that goal. Such an example is when we are writing a letter. We focus on the contents of the text, not on the correct application of grammatical rules. We do not have to think about them, we are able to use them automatically. Talented people are those who constantly practice their talents and enjoy doing so.[2] The identification and nursing of talents from an early age is highly important. To properly help in unfolding and developing above average human properties, early identification and talent care through a personalized requirement system is of utmost importance. The current technological momentum indicates that by the time the secondary schoolers of today reach the 2030s, the demand for advanced programming capabilities will rise rapidly, even as much as 90%. For this reason, early identification of people with such capabilities will be important. [3]

### *B. Factors of age-independent examination*

During our earlier research work, we created a software solution that does not require capabilities which young children may not exhibit yet. When creating it, we put emphasis on taking those skills into account that can develop parallel to algorithmizing but do contain some elements which we are born with as well. An example of such is number sense, which is the congenital foundation of calculating capabilities and is almost completely decoupled from other cognitive abilities.[4] This helps us recognize quantitative changes in sets of smaller cardinalities, as well as proper determination of magnitude differences. Beside number sense, we have something relatively similar called sense of space, which helps us understand the places occupied by objects, as well as their changes in position through movement. Memory has an effect on the creation of algorithms, as well as on our calculation and spatial awareness capabilities. Our working memory can only process a limited amount of information at a given point in time and the storage of those is transitory (a few minutes or just seconds). However, it is a key element when acquiring new skills and knowledge. Its level of development influences (and at kindergarten age, it projects) the performance exhibited in school as well. Also, when it is damaged, it can be brought into connection with certain abnormalities, for example ADHD (Attention Deficit Hyperactivity Disorder).

## III. TEST SYSTEM PRESENTATION

Taking into account the aforementioned aspects of age-independent examinations, we created a test system based on light programming. The main element of the system is a moving pattern which is constructed from and presented by a series of lights. The test subject has to break this pattern down into its individual elements and “program” them, which means that they have to build a sequence of steps that resembles the original pattern as closely as possible. To achieve this, the creation of an algorithm is required, which breaks down the constant changes in position into steps that follow each other in a given order, then returns these steps.

Afterwards, based on the various properties of the returned sequence, we can deduct conclusions about the level of development of the subject’s algorithm creation capabilities. The greatest advantage of this approach to implementation, versus other methods available, is that it does not require any kind of preliminary programming knowledge.

The test system is based on eidetic (visual) memory and binary thinking. The goal is, that the test subject can convert the "pictures" stored in their memory to a binary signal system, in correspondence with the LEDs' emitting and dark states, thus algorithmizing the individual steps.

Before facilitating the testing procedure, we provide clear instructions to the supervising educator, in order to ensure that the tests are filled out completely and correctly. The following step is an anonymous survey which the test subjects have to

fill out, stating their gender, age, previous programming experience etc. This will be needed at a later point when we are compiling statistics.

After filling out the survey, we play a pattern for the test subjects, in which we differentiate LEDs that are in a lit up or turned off state. Subjects have to reproduce the sequence of these LEDs lighting up. For each test case, they receive 8 distinctive light patterns. During the resolution of one pattern, we evaluate certain parameters, which we are also logging. These parameters include for example the number of clicks, the number of steps required for the solution, the number of steps used by the test subject for the solution, the elapsed time as well as the number of right and wrong states.

Figure 1: The representation of LEDs lighting up on the webpage

## IV. MEASUREMENT RESULTS PRESENTATION

Using the test system, we have conducted measurements in various educational facilities across the city of Debrecen.

| | **Under 6** | **6-10** | **10-14** | **14-18** |
|---|---|---|---|---|
| Capita | 15 | 71 | 81 | 118 |

Figure 2: Number of test subjects grouped by age (years)

As can be read from Figure 2, we were able to conduct our tests with multiple age groups.

When calculating the correct answers, we took three cases into consideration. If the subject entered exactly as many steps as there are in the example pattern, then we calculate how many of these steps were switching LEDs correctly. If the number of steps entered is more than what is required, then we considered the first n states and the correct LED settings within those. If someone saved less steps than required, we evaluate all of them from a correctness perspective. As a result, we acquire how many LEDs were set correctly during the 8 tests overall. Then, if we took this value and divided it by the sum of the incorrect settings, we could determine the percentage of success for each individual.

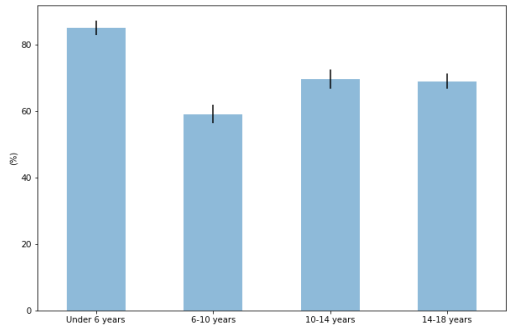


Figure 3: The average performance of all age groups (%)

Based on the achieved results, it can be clearly observed that children of kindergarten age (under 6 years) performed exceptionally well during the measurements, which proves that you can already examine algorithm creation capabilities at these early ages. In Hungary, children learn more and more IT related subjects as they progress to higher grades, with the oldest ones learning programming as well. The above diagram also indicates that individuals who have received more and deeper content through informatics education performed better in our tests.

Going further, we would have liked to examine the average number of correct LED settings in each age group through the lens of gender.

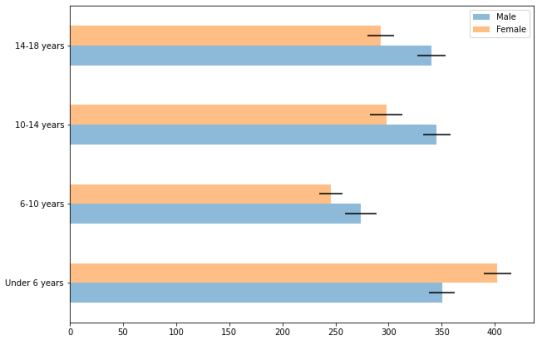


Figure 4: Number of correct LED settings grouped by age and gender

With the help of Figure 4, we can deduct that we did not experience a significant difference between the number of correct answers of male and female participants in the kindergarten and young primary school age groups. This is because at these ages, the opportunities for differentiated learning are very limited. However, in the 10-14 age group and secondary schoolers, a greater disparity between male and female results can be observed. It

can be said that boys spend more time with such activities on average. Those individuals who have more experience and practice, generally performed better on our tests. This is clearly reflected in the results belonging to the 14-18 age group. The performance between genders does not show a significant difference.

## V. Analysis of Results Using Convolutional Neural Network

Machine learning and deep learning are some of the most powerful statistical learning tools in data science. With these, we can find non-deterministic patterns and we are also able to fit non-deterministic state transition functions. These are very good but not good for every situation. If there is a trivial way to solve one problem, it's recommended to not use statistical learning algorithms.

In our case, there will be a trivial function to predict any metrics about children, but we have two reasons to not find it. We want to examine that can we predict their ages through the test that we developed, and it can also help to deterministic any kind of mental degeneration. The other reason is we want to focus on their privacy and anonymity also. The data is handled as private information about the children, but it could help the parents in form of online tests or games on their phones. And the programs with it do not need any private information, just the result of our test.

The first thing that we have to define is the main plan of artificial intelligence. The goal of intelligence software is to imitate human choices in real situations. All software can be defined as intelligent software. The main difference is the way when they find the state transition functions.

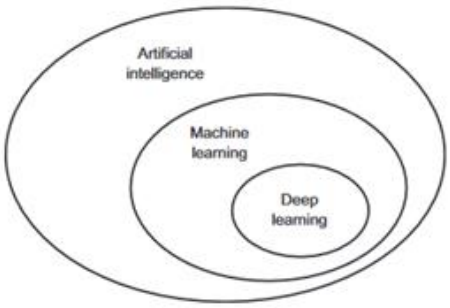


Figure 5: Relation between artificial intelligence, machine learning and deep learning

The traditional intelligence software is hard coded, which means the programmers give the rule between two states. These programs get to their input the rule and the input data and with these pieces of information, the program produces the output.

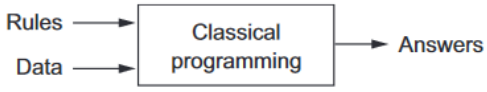


Figure 6: Classical programming

The learning algorithms do not need explicit rules to define state transition functions between two states. Instead, we give a model with controllable parameters, which are named weights. In this case we know the input an the expected output, and we want to find the state transition function. In the learning process, we measures the distance between the model and the expected results, and about this value we set the weights of the model.

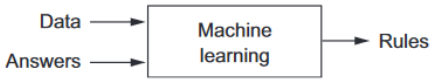


Figure 7: Learning algorithms

To learn the weights or parameters of the model is used an optimizer algorithm is. This algorithm tries to find the optimal minimum point of the weights' cost function. The cost function measures the distance between the expected output and the model's actual output, under this process the optimal weights are found but often we have to iterate more then once.

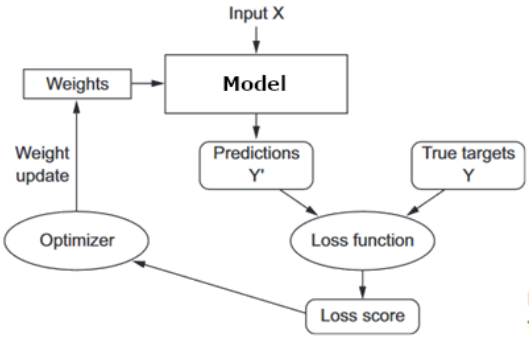


Figure 8: Learning process

All of the learning algorithms are part of the topic of machine learning. A machine-learning model tries to transform its input data into meaningful outputs. For example, if we want to classify images that consist of cats and dogs, the meaningful output is the class of the current image. We can think of it as the network tells this is a cat or this is a dog. Deep learning is a specific subfield of machine learning. The deep reference to layers of representations. The model in the case of deep learning consists of layers. How many layers contribute to a model of the data is called the depth of the model. In deep learning, these models called neural networks, are structured in literal layers stacked on top of each other.

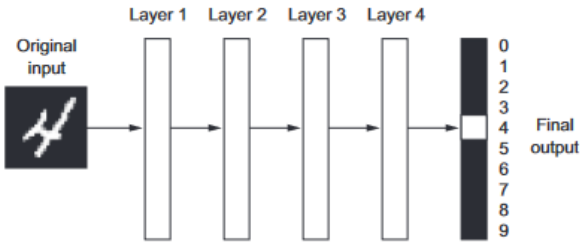


Figure 9: An example of working network

The neural network works similarly to an assembly line. The current layer gets the other layer's output as input data. One layer is doing an operation in the data and gives the processed data forward in the network. We can choose these layers from a wide range like convolution, dense of fully connected, LSTM, and so on.

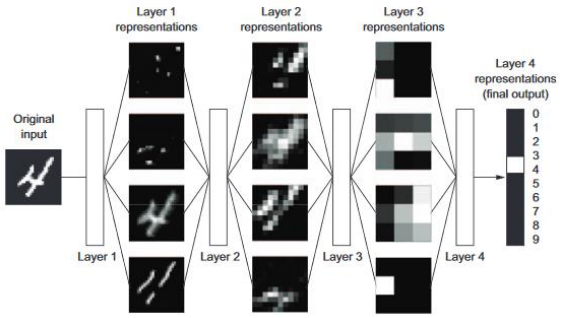


Figure 10: Inside of a neural network

The network tries to find the relation between the input data and the expected output by the learning process. The working of the network can be controlled by the cost function. The big bottleneck of the network is a bad cost function. This function gives a rate about the output of the network, and it means the distance from the expected output. The learning process wants to minimize the cost and it shepherds the weights for better values. With special cost functions, we can approximate probability distribution and their density function also. With a good cost function, we can give meaning to severe numbers.

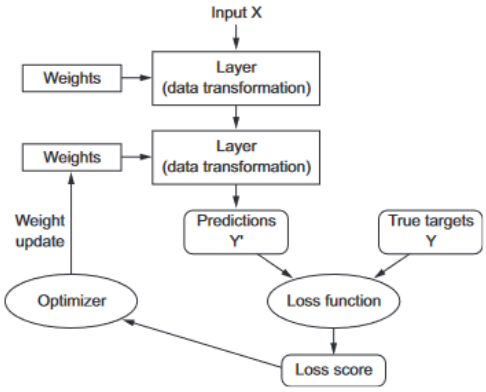


Figure 11: Learning of neural networks

There are some problems that we should understand with neural networks. One other bottleneck is the architecture of the model. A good model highlights the important features and parts of data, but too complex a model can be too specialized for the training data, which is called over fitting. And the big network absorbs the inside information of the data. A too-small model is generalized too much and distorts the information. There are architectures that have been developed for many

problems like image recognition. There are no universal architectural planning patterns. Not every architecture is good for every task.

A machine-learning algorithm can do two types of tasks: classification or regression. A regression problem is when we want to get a continuous output value like the age of a person in an image. A classification problem gives a discrete output about the classes, like when we want to tell the number of the class in an image. In the case of a classification problem, we get a vector as a result, and an element is a distribution value. This value tells us what is the possibility the current data is part of this class.

We planned our architecture by analyzing the results. We based it on a very simple convolutional network. The convolution is good for finding patterns inside the data. Under the convolution operation, we understand in case of $A$ nm and $K$ $n_1m_1$ matrices , $a_{ij} \in A$; $a_{ij} \in R$; $k_{ij} \in K$; $k_{ij} \in R$

$$\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} * k_{ij}$$

In the case of this layer, there can be $I$ convolution filters in one layer which means the actual layer wants to find $I$ piece pattern and we have $K_i$, i = 1,2,...,$I$. filters. These filters are moved in case of images from top to bottom, left to right. We used a one-dimension version of the convolutional layer, where the data $A$, $K_i \in \mathbb{R}^2$ are vectors. $K_i$ $ filters are learned under training.

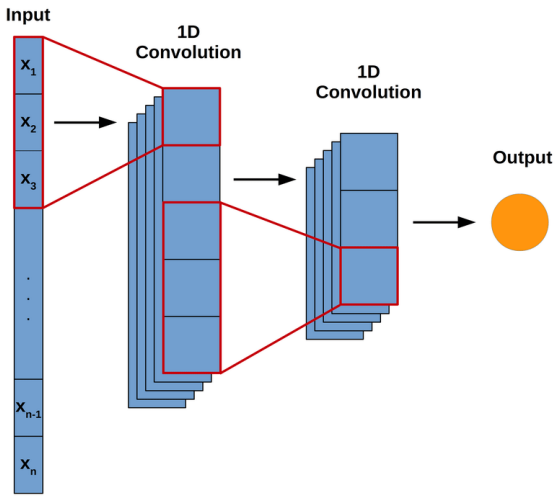


Figure 12: One-dimension convolution

The network can be separated into two parts. One is the embedding and the other is the classifying or regressing(depending on our problem) subnetwork.

The convolution layer creates multidimensional matrices by the different convolution filters. We have to flatten the embedded data into a single vector. It means we concatenate all dimensions into one dimension.

The convolution layers do the feature embedding and highlight the main information. As result, we get the data in a new representation of an embedded vector, which alone is unusable. To draw a conclusion about the embedded vector we have to use another layer, the fully connected or dense layer. These layers are based on multi-variable linear regression. That means, inside these layers the operation is:

$$g(\sum_{j=0}^{n} w_i * x_i)$$

where $w$ is the weight of the layer, $x$ is the vector from previous layer. $g$ is an activation function, that gives non-linearity to the model and extends the hypothesis space.

In Table one, you can see the model we used. This is a very simple model because we have very tight information in the data. A large model absorbs the information from the data. The convolution is necessary to use to recognize the relevant patterns.

For training our network uses Adam optimizer. For the convolution layers, we used ReLU activation. For the regression, we do not have activation.

Table 1: Model summary for test.

| Model | | |
|---|---|---|
| Layer (type) | Output Shape | Param # |
| Conv1D: 16 filter | (None, 92, 16) | 48 |
| Conv1D: 8 filter | (None, 91, 8) | 264 |
| Flatten | (None, 728) | 0 |
| Dense 93 unit | (None, 93) | 67797 |
| Output | (None, 1) | 94 |

Total params: 68,203
Trainable params: 68,203
Non-trainable params: 0

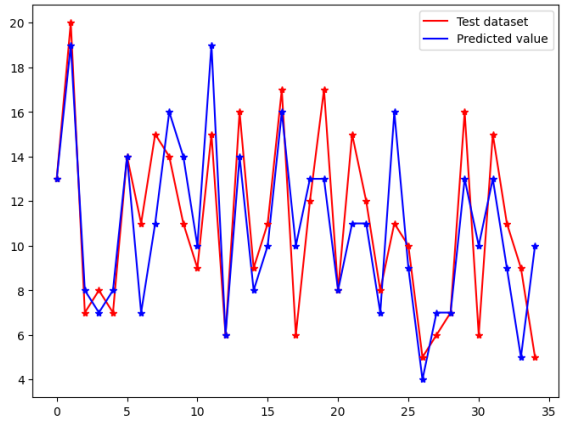


Figure 13: One-dimension convolution

The Figure 13 you can see the result of the network on the test data. Many times our network forecast the ages very close to the real value, which means the children do not have any mental illnesses. In some cases, there are big errors between the predicted value and the real value which means the current children could have mental problems if the predicted age is lower than expected and it is recommended for examination by a doctor or a therapeutist because the child can be even tired or has any other problems which are temporarily only. If the predicted value is higher than the expected means that the child does not have any problem.

## VI. SUMMARY

In addition to flexibility of thinking and the level of efficiency in problem solving, the efficiency of computing, calculating and algorithm creating processes are key elements of the various capabilities and skills that influence programming. These can have a filtering effect on each other. The basis of the test system that we developed is light programming, which can remarkably decouple the measurement results from the subjects' age and level of programming education.

The algorithm creating capabilities greatly influence the successfulness of education, therefore its deeper research is properly justified. A better understanding of how algorithm creating capabilities are formed can bring us closer to improving the efficacy of programming education from primary school all the way to university.

It is important that children learn programming in their IT classes, since our measurements clearly showed that individuals with previous programming experience performed much better in the light programming exercise as well.

We can deduct several conclusions from our measured parameters, which can help in filtering the talented students, thus giving the ability to guide them towards a programming career by making it interesting for them, which would ultimately help alleviate the shortage of programmers on the global market in the future.

Going forward, we would like to further develop and enhance the platform. We would like to examine more parameters, for example the level of education of the parents, to see if we can find any correlations between a subject's performance in the light programming test and the highest educational attainment of their parents. It is also among our plans to expand the multilingualism of the platform. This would open opportunities to compare the results of our local students versus students from other countries.

## REFERENCES

[1] Amorim Cl´audio, Beyond Algorithmic Thinking: An Old New Challenge for Science Education, Eighth International History,Philosophy, Sociology & Science Teaching Conference, University of Leeds, England (2005)

[2] Subotnik, R. F., Rethinking Giftedness and Gifted Education: A Proposed Direction Forward Based on Psychological Science, Psychological Science in the Public Interest, 12(1), 2011, Pages 3–54.

[3] Ali C¸etinkaya, ¨Omer Kaan Baykan, Prediction of middle school students' programming talent using artificial neural networks, Engineering Science and Technology, an

International Journal Volume 23, Issue 6, December 2020, Pages 1301-1307.

[4] Kaufman, E., Lord, M., Reese, T., & Volkmann, J., The discrimination of visual number, American Journal of Psychology, 62, 1949, Pages 498-525.

[5] G. White, M.A. Sivitanides, Theory of the relationships between requirements of computer programming languages and programmers' cognitive characteristics, J. Inf. Syst. Educ., 14 (2003), Pages 409-416.

[6] J. Bennedsen, M.E. Casperse, Revealing the programming process, ICER Proc., 16 (2015), 10.1145/1047344.1047413, Pages 155-163.

[7] J. Zagami, M. Boden, T. Keane, B. Moreton, K. Schulz, Girls and computing: female participation in computing in schools, Aust. Educ. Comput. 30 (2015) 2.

[8] J. Bughin, E. Hazan, S. Lund, P. Dahlstr¨om, A. Wiesinger, A. Subramaniam, Skill shift: Automation and the future of the workforce, McKinsey Global Institute 1 (2018), 3–84.

[9] Israel, M., Pearson, J., Tapia, T., Wherfel, Q., Reese, G., Supporting all learners in school-wide computational thinking: A cross case analysis, Computers & Education, 82, 263-279 (2015).

[10] Marino, M. T., Gotch, C. M., Israel, M., Vasquez, E., Basham, J. D., Becht, K., UDL in the middle school science classroom: Can video games and alternative text heighten engagement and learning for students with learning disabilities? Learning Disability Quarterly (2014), 37, 87-99.,

[11] Q. Burke, Y.B. Kafai, Decade of game making for learning: From tools to communities. Handbook of digital games, (2014), pp. 689-709.

[12] J. Good, Learners at the wheel: novice programming environments come of age, International Journal of People-Oriented Programming, 1 (1) (2011), pp. 1-24.

[13] I. Harel Caperton, Toward a theory of game-media literacy: playing and building as reading and writing, International Journal of Gaming and Computer-Mediated Simulations (IJGCMS), 2 (1) (2010), pp. 1-16.

[14] Y.B. Kafai, Q. Burke, Connected code: Why children need to learn programming, MIT Press (2014).

[15] L. Lambert, H. Guiffre Computer science outreach in an elementary school, Journal of Computing Sciences in Colleges, 24 (3) (2009), pp. 118-124.

[16] J. H. Borland, Gifted children without gifted education, In Sternberg, R. J., Davidson, J. E. (Eds.), Conceptions of giftedness (2nd ed., pp. 1–19.). New York, NY: Cambridge University Press (2005).

[17] D. Shenk, The genius in all of us: Why everything you've been told about genetics, talent, and IQ is wrong, (2010) New York, NY: Doubleday.

[18] Y.J. Lee, Empowering teachers to create educational software: a constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship, Computers & Education, 56 (2) (2011), pp. 527-538

[19] J.M.C. Lin, L.Y. Yen, M.C. Yang, C.F. Chen, Teaching computer programming in elementary schools: a pilot study, National EducationalComputing Conference (2005)

[20] R.E. Stake, Multiple case study analysis, Guilford Press, New York (2006)