International Journal of Child-Computer Interaction 31 (2022) 100415

Contents lists available at ScienceDirect

International Journal of Child-Computer Interaction

journal homepage: www.elsevier.com/locate/ijcci

Research paper

Teaching machine learning in elementary school

Gilad Shamir *, Ilya Levin

Tel Aviv University, Tel Aviv, Israel

ARTICLE INFO

Article history:
Received 30 December 2020
Received in revised form 9 September 2021
Accepted 13 September 2021
Available online 23 September 2021

Keywords:
Computational thinking
Constructionism
Artificial intelligence
Machine learning
Elementary school
Programming

ABSTRACT

The emergence and ubiquity of Artificial Intelligence in the form of Machine Learning (ML) systems have revolutionized daily life. However, scant if any attention has been paid to ML in computing education, which continues to teach rule-based programming. A new, promising research field in education consists of acquainting children with ML to foster this much-needed shift from traditional rule-driven thinking to ML-based data-driven thinking. This article presents the development of computational thinking competencies in 12-year-old students who participated in a learning-by-design or a learning-by-teaching ML course. The results, based on a qualitative and quantitative evaluation of the students' achievements, indicate that they demonstrated computational thinking competencies at various levels. The learning by design group evidenced greater development in computational skills, whereas the learning by teaching group improved in terms of computational perspective. These findings are discussed with respect to promoting children's problem-solving competencies within a constructionist approach to ML.

1. Introduction

Societies and education are undergoing radical changes as a result of what is known as the digital turn (Levin, & Mamlok, 2021). These changes impact our daily lives, but also raise fundamental questions about our understanding of the world around us. Preparing the younger generation for life in the new digital world is of crucial interest to educators, and in particular the best ways to enable elementary school students to master emerging technological phenomena. Students can and should understand not only the basics of interactions with digital reality, but also its construction and ontological principles. The use of ML systems in education has grown exponentially in the last few years, mainly due to the increased availability of ML resources (Domingos, 2015; Druga, Vu, Likhith, & Qiu, 2019). The current study reports on a pilot experimental program to foster students' computational thinking (CT) skills as they develop through the implementation of ML systems in elementary school.

This paper employs several key concepts that are widely used in both the mass media and academic sources. Because these concepts are often ascribed different meanings, it is crucial to define them at the outset in terms of the context in which they are used in this article.

• Computational Thinking (CT) can be defined as the thought processes involved in formulating problems and their solutions in ways that a computer could also execute (Wing, 2006). The problems can be tackled by an information-processing agent which can be a computer or a human. CT attempts to solve complex problems by implementing a three-step problem-solving process: (1) decomposing the problem into sub-problems, (2) solving the sub-problems, and then (3) combining the solutions to the sub-problems into an overall solution (Leiser, 1996). This process requires the use of competencies such as decomposition, pattern recognition, abstraction, and algorithm design. These are now widely accepted as comprising CT and form the basis of curricula that aim to support its learning and assess its development worldwide (Grover, 2017), and in particular in Israel (Ministry of Education Israel, 2021). They can be referred to as "traditional CT" or "automation CT" competencies because they are used in the cognitive effort of automating a solution to a problem. It is worth noting that the computing education community has yet to find a consensual definition of CT (Tedre, & Denning, 2016).

• Artificial Intelligence (AI) systems are computer systems that can perform tasks that would normally require human intelligence (McCarthy, Minsky, Rochester, & Shannon, 2006). AI systems are characterized by their ability to learn. The data generated by an AI learning process is used to make decisions akin to those of humans in similar situations. This is often framed by considering AI as the computer's ability to create new knowledge from data and use that knowledge to make human-level decisions (Rouhiainen, 2018).

• An Artificial Neural Network (ANN) can be viewed as an AI system that simulates the interaction of human neurons in the brain.

* Corresponding author.
E-mail addresses: giladshamir@mail.tau.ac.il (G. Shamir), ilia1@tauex.tau.ac.il (I. Levin).

https://doi.org/10.1016/j.ijcci.2021.100415
2212-8689/

An ANN can be represented as a graph consisting of connected neurons as binary elements, in which outputs of elements are connected to inputs of others (Yegnanarayana, 2009).

• Machine Learning (ML) is one way to implement AI. The first personal computers were programmed to run predesigned algorithms. In contrast, thanks to ML, today's computers and their digital artifacts can accumulate experience and data and improve their functioning based on this data to become self-evolving AI systems.

• Deep learning is a kind of ML. It is based on multilevel neural networks consisting of multiple layers of artificial neurons. The development of deep neural networks is considered one of the most important advances in digital technologies in recent years. These technologies have made it possible to come very close to achieving human-like visual perception on recognition tasks. This fact in itself is unprecedented and remarkable. The traditional concept of a computer's capabilities simply viewed the computer as the executor of a predesigned rigid algorithm. Modern digital devices equipped with deep learning capabilities have changed the traditional paradigm of computing significantly. Computers can now not only execute a program written into its memory in advance but also learn and develop from experience.

Both academia and the popular media have taken growing interest in ML, its characteristics and its social impact. However, some crucial issues remain relatively unexplored although they are essential for understanding processes taking place in society, especially with respect to education. One of these neglected issues, which in fact prompted this study, is the change in the concept of computing itself. The fact that a computer with ML abilities no longer merely carries out the program stored in its memory has had major ramifications for the concept of CT and how it is taught. The second key insight motivating this study is that one of the most important contributions of ML has gone mostly unnoticed: it can supplement human intelligence (Domingos, 2015). Thus the ways in which ML can be integrated into education is both fascinating and of critical importance. Specifically, how has the emergence of ML affected current computing school curricula and what effects will it have on students' knowledge, skills, and perspectives? To answer these questions, we explored the feasibility and effectiveness of introducing ML in elementary school, not as a new stand-alone discipline, but as a way to expand and enrich existing fields, based on CT.

The idea to include ML education as part of CT learning has wide and justified support (Mariescu-Istodor, & Jormanainen, 2019). The current study constitutes a step towards achieving this goal. Below we describe ways to develop elementary school students' competencies to construct ML systems, inspired by the Constructionist Approach. The choice of this pedagogical approach is related to the history of Constructionist ideas. The founders of Constructionism had a ML research background. Seymour Papert (Papert, 1980) and his followers considered that there is a profound interconnection between human learning and machine learning (Badie, 2016; Kahn, & Winters, 2020). Today this interrelationship continues to elicit considerable scientific and practical interest (Levin, & Tsybulsky, 2017).

In Shamir, and Levin (2021) we presented an experimental ML course and its outcomes in terms of students' motivation to learn and to understand the fundamentals of ML. Results showed high engagement during constructionist learning and that the novel programmable learning environment, SingleNeuron, helped make machine learning understandable. Using the perceptron mechanism allowed students to create their own ML-based artifacts and explore a wonder of AI: how one artifact satisfies different purposes when trained with different datasets. This paper further develops the study and compares CT gains of two different ML courses based on a CT-ML framework that we designed to study and evaluate elementary school students' CT development while constructing ML systems.

The remainder of the paper is organized as follows. The theoretical background is provided in Section 2. Section 3 deals with the methodology, research design, and procedure. The results are presented in Section 4 and the discussion in Section 5.

2. Background

Machine Learning (ML) has changed practically all the areas of our lives, from health care to politics to journalism. Whereas the Industrial Revolution automated manual work and the Information Revolution did the same for cognitive endeavors, ML has automated automation itself. When working in the traditional Automation paradigm, a software developer intends these systems to be internally controlled and self-regulated. In other words, humans who design programs set down rules for the system agents and their interrelations. During runtime, when the software executes, these agents implement the program with no further human intervention. However, this is not the case in the automation of automation paradigm. Rather, the system of agents changes the system's behavior autonomously in a way that its creator is not aware of and cannot predict. In such cases, once the program is executed, there is no human intervention, and the behavior of the agents changes with no outside intervention (Domingos, 2015).

One of the most exciting aspects of building ML systems is that it challenges the traditional computational thinking (CT) skill set. ML does not follow the three classical CT problem-solving steps mentioned above. In this sense, ML makes problem-solving different and can redefine CT's educational goals. If indeed ML is the automation of automation, what are the CT skills an individual needs to develop an "automation of automation" solution to a problem? How can these skills be imparted to elementary school students?

Here we harnessed the Constructionist approach to teach ML in elementary school. Constructionism is usually referred to as learning-by-design. It applies to all domains, and suggests that students learn best when creating and using external representations for modeling and reasoning (Blikstein, & Wilensky, 2009; Papert, & Harel, 1991). The Constructionist approach emerged from the famous and widely known Constructivist approach (Ackermann, 2001). Papert was influenced by Piagetian principles and many of Jean Piaget's Constructivist ideas were adopted by Constructionism. However, Constructionism also enriched Constructivism with remarkable and important innovations. These innovations relate primarily to the emergence of digital technology in the 1980s and include the fact that learning-by-design involves a new kind of activity; namely, designing software artifacts. Papert realized that program/algorithm design, which includes software development, would be similar in many ways to teaching in the upcoming digital era. This realization was rooted in the nature of CT itself, which requires the learner to formulate rules for the behavior of an artifact while in the process of creating it. Since then, the Constructionist approach has gone beyond traditional learning-by-design to encompass the new component of learning-by-teaching, a computer metaphor (Vartiainen, Tedre, & Valtonen, 2020). As ML has gradually penetrated educational practices, this component of learning activity has become more prevalent. This can be seen in the transition to data-driven learning from the rule-driven learning that characterized traditional programming. This trend is also reflected in the structure of this study, which is based on observations of students in the two different courses developed for our research purposes. The first course involved designing an artificial neural network and was

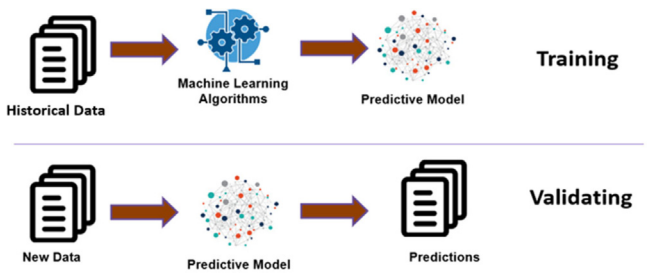

Fig. 1. The two-step machine learning process.

entitled "Learning ML by Design". The second course involved applying data to train a ML system and was entitled "Learning ML by Teaching".

One of the important issues when conducting research on innovative ideas and methods is the choice of the computer environments used by the students. Numerous studies have concentrated on designing introductory programming environments that make programming more accessible to younger learners as well as more fun and engaging (Brennan, & Resnick, 2012). A popular approach in recent years is visual programming environments that provide visual blocks as connected commands. This approach is an effective way to allow novice learners to experience early programming successes. However, a general-purpose programming environment has its flaws. The major drawback is that it has endless opportunities, and learning to use it is a time-consuming process, making it impractical for across-the-board use in the general subject matters taught in school (Jona et al., 2014). To overcome this shortcoming, researchers have created domain-specific micro-worlds based on a general-purpose programming environment with a dedicated set of blocks for the students to use. Here, to scaffold the learning activity of constructing a simple artificial neural network (ANN), we developed a Single-Neuron toolkit (Shamir, & Levin, 2020). Our curriculum also included additional platforms, as described below.

ML endows computer systems with the ability to learn without being explicitly programmed to solve a predefined problem. In ML, the computer software is programmed to use example data to solve a given problem. Once programmed, the system is trained on given data. The trained ML model can later be used to predict a future event's probability within acceptable reliability.

There are different ways to describe the processes of ML construction. One is the two-step Train-Validate process that requires some intermediary actions (Guyon, & Elisseeff, 2006; Khalid, Khalil, & Nasreen, 2014; Shamir & Levin, 2020; Zimmermann-Niefield, Turner, Murphy, Kane, & Shapiro, 2019). Specifically, in the training step the programmer needs to apply appropriate data preprocessing such as data acquisition, feature selection and data splitting. These make the dataset suitable for the machine to learn from. The next stage involves selecting an algorithm and activating it on the preprocessed dataset. In the validation step (also known as predicting or scoring) the model is tested to maximize the predictive performance of the final system. This requires establishing a validation dataset that was not part of the training dataset, predicting the result for each data entity running the system, and validating the system's predictions. Fig. 1 illustrates both steps.

In this study, we used a traditional CT framework to study and evaluate CT (Brennan & Resnick, 2012) with the following key dimensions:

(1) Computational Concepts: the terms and expressions practitioners engage with as they create computational artifacts.

(2) Computational Practices: the techniques and abilities practitioners develop as they engage with the concepts.

(3) Computational Perspectives: the perceptions practitioners form about themselves and the world around them.

We added a taxonomy for creating ML systems to this CT framework to study and evaluate elementary school students' CT development while constructing ML systems, termed the CT-ML framework. Since the ML process consists of the training and validating steps, the computational practices dimension was divided into two sub-dimensions: Machine Training Practices and Machine Validating Practices. A diagram of the CT-ML framework is presented in Fig. 2.

In Fig. 2, the orange items are computational ML practices. The green items are the computational perspectives reflecting how the practitioners perceive themselves and their surroundings when constructing ML artifacts. In gray are the computational concepts, which are what practitioners engage with as they create ML artifacts (Guyon & Elisseeff, 2006; Khalid et al., 2014; Zimmermann-Niefield et al., 2019).

Each computational practice element of the taxonomy is detailed below. Applied examples appear in the Results section.

- Category selection: The ability to find similarities to create prototype examples of categories.
- Data selection: The ability to choose which data to acquire that is most likely to improve the system's prediction ability and avoid bias. This requires considering the different features of the object to be labeled and selecting a diverse dataset.
- Data split: The ability to separate a set of data into two groups: one for training and one that is held out for validation.
- Data filtering: The ability to remove specific types of data before the training phase to improve the system's classification capability.
- Feature selection: (also known as feature extraction): the ability to find only those attributes that contain information pertinent to the system's data classification.
- Prediction: the ability to determine the likelihood of an ML model's outcome after it has been trained on a historical dataset and then applied to new data.
- Evaluation: the ability to assess how well the system can categorize the validation dataset and determine how the results can be improved to enhance classification. Evaluation covers such issues as whether the machine was undertrained, whether the model was biased towards a certain object, whether the features were appropriately selected and whether the dataset was split appropriately into train and validate.
- Algorithm creation: The ability to create the set of steps that will be used by the system's learning mechanism.

Our vision of ML literacy involves stakeholders participating in ML construction. Given that computational practices for creating ML systems and traditional systems differ, it is useful to view ML practices as the expansion and development of conventional CT practices based on a Constructionist pedagogy. These expansions are listed in Table 1. Note that the four traditional computational practices deal with the process of formulating the problem and creating a solution, and not with the evaluation of the solution;

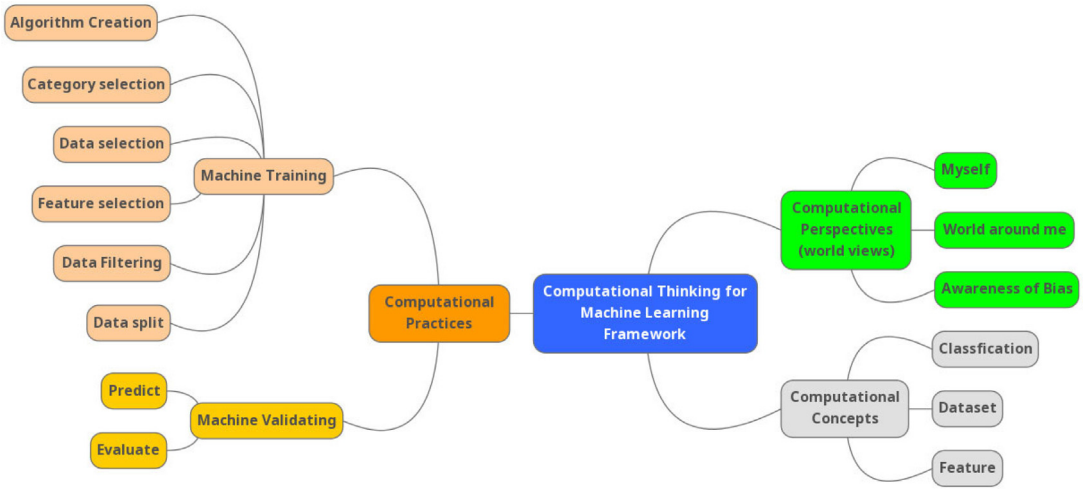

Fig. 2. A diagram of the CT-ML framework.

Table 1
Traditional CT practices vs. ML CT practices.

Traditional CT practices	ML CT practices
Decomposition	Data split: The data is decomposed by splitting the dataset into a training dataset and a validation dataset, Feature Selection: To identify the features in a complex data entity such as an image, it is useful to initially break down the data item into subparts.
Pattern recognition	Category selection: Involves classifying the training dataset into the problem categories the ML will be trained to recognize. This requires recognizing a pattern. Data filtering: To use data that are relevant to the classified categories, the patterns relevant to the anticipated features need to be identified. Predict: The need to predict the category of a data entity in the validation dataset based on features.
Abstraction	Feature selection: For a ML model to properly classify the data entities, only those input dimensions that contain the relevant information for solving the problem must be selected.
Algorithm creation	Algorithm creation: The machine model is based on a neural network algorithm which needs to be either programmed or given.
	The following are ML CT practices unrelated to PRADA practices. Evaluate: Evaluating the verification process requires comparing the classification prediction to the machine's classification. Data selection: To reap the benefits of ML, decision biases must be minimized.

therefore, there is no matching transition skill to "Evaluate" in the table. The traditional CT practices listed are those defined by the Ministry of Education in Israel, but other traditional CT frameworks do have some form of 'Evaluate' such as the Operational Definition of Computational Thinking created by CSTA (International Society for Technology in Education & Computer Science Teachers Association, 2011).

The generally accepted opinion is that the bulk of CT is specifically aimed at figuring out how to get a computer to do a job for us, while algorithms are the procedures that specify how the computer should do this (Denning, & Tedre, 2021). Shifting from rule-driven to data-driven artifacts may thus reorient the focus of CT. It raises the following essential questions with regard to embedding ML in CT curriculum: How do traditional Computational Thinking and innovative Machine Learning thinking relate? Is ML part of CT or an extension of it? Does our concept of CT change after integrating ML components?

In this section, we presented a possible response to these questions in the ML-CT framework. In addition, we discussed the traditional CT practices and the possible associations between them and ML CT practices. The next section describes the research design implemented to assess ML CT.

3. Research design

This study investigated the acquisition of the fundamentals of ML in elementary school. Specifically, we examined students' computational concepts, practices and perspectives around a set of learning modules involving two different computational platforms and curriculum.

In his book entitled the Dual Nature of Technical Artifacts *(Du mode d'existence des objets techniques*, 1958), the French technological philosopher, Simondon, considered the genesis of technical artifacts to be the object of human knowledge (De Vries, 2008). According to Simondon, it is not enough to focus on working with an artifact; rather a person's knowledge of artifacts should be based on theoretical insights into their functionality. Consistent with this notion we believe ML learning should include constructing an ANN. We designed a computer-based learning environment called Single-Neuron to be used in conjunction with ML learning. Single-Neuron is an ML modeling toolkit that combines general computational primitives and domain-specific primitives. The primary mode of interaction with this modeling toolkit is through code. Single-Neuron makes it easy for a learner to write a program within a short time, and diverse outcomes can be observed from a small set of rules. It uses a block-based interface and is created using Scratch (Resnick et al., 2009).

Introducing the ANN construction activities constitutes a major innovation in elementary school. The Single-Neuron toolkit is used for scaffolding the ANN creation activity, and a corresponding learning module was developed for this purpose. The learning by design approach was implemented in one course therefore it is called "Learning ML by Design". The other course was entitled "Learning ML by teaching" and involved using a preexisting ANN to construct an ML system. In that course, the students created their own categories, their own training dataset, and validation set, which they used to train the machine and validate its predictions. These two courses formed the pedagogical framework of our study. The results section compares the outcomes of these two learning approaches in terms of students' CT gains.

3.1. The ML curriculum

To study students' acquisition and development of CT ML competencies, we developed four learning modules that actively engage students with ML using a variety of learning platforms. The modules were: (1) Introduction to ML, (2) Practicing the ML process, (3) Constructing a data-driven ML system and (4) a Constructing rule-driven ML system.

The modules comprising the ML curriculum were administered in two different courses. The Learning ML by Design course consisted of modules 1, 2, and 4, and thus focused on all aspects of ML, including creating an Artificial Neural Network (ANN). The second course, Learning ML by Teaching, consisted of modules 1, 2, and 3, which enhanced training and validating an ML system but omitted the ANN construction.

The module activities were based on the Use-Modify-Create learning progression model which facilitates learning using scaffolds. It consists of a three-stage progression for engaging in CT within computational environments. It is based on the premise that scaffolding increasingly deep interactions promotes the acquisition and development of CT (Lee et al., 2011). Each learning module focused on one progression level of the model, as illustrated in Fig. 3.

Module 1 — Introduction to ML

In this module, the students were involved in discovering how ML works and the ways in which it differs from rule-based programming. They were given traditional algorithm creation tasks to demonstrate the uniformity of their results in rule-driven computing. As a contrast, they were shown examples of ML mistakes. While doing so they were introduced to data-driven computing. The students were also given an opportunity to engage in a conversation with an AI chatbot and were encouraged to decide whether it passed the Turing test. This module used the Code.org platform for algorithm creation and the Mitsuku website for AI conversation. In this module the students' CT progression level was the initial 1 – 'use', and they did not program ML or train a machine.

Module 2 — Practicing the ML process

This module advanced students from ML users to modifiers, which is the next level in the CT learning progression model. The students carried out a set of training activities on an ML system that involved classifying images into two categories. The categories increased in complexity as the activities progressed. They required the students to train and validate the classifier model. The context was ecological systems, based on the 6th grade science curriculum in Israel. The ML platform used was "AI for oceans" by the Code.org organization, which consists of a ready-made ML algorithm and a given ML training interface.

Module 3 — Constructing a data-driven ML system

In this module the students progressed from modifiers of pre-made ML systems to actual creators, which is the top level in the use-modify-create CT learning progression. They were required to train a ready-made ML machine and program the software that served as its client. The participants chose what the ML would classify; in other words, they decided which categories to form. Module 3 had more requirements than in Module 2, which only had pre-made categories. The students also created the text-based dataset to train the machine. The client software programming was done using a dedicated programming interface to provide interactions with the pre-made server. Finally, the students activated their software, using a validation dataset of their own to evaluate the machine's performance with respect to their predictions. The module used the Machine Learning for Kids platform, which provides an interface for users to define labeled buckets and populate them with text. The module also uses the Scratch micro-world.

One student's project in Module 3 was a ML system that determines whether a person is a Minecraft fan or a Fortnite fan based on inputting a statement. The student selected the Fortnite and Minecraft categories and created the training datasets, which consisted of statements related to each category. The training set is shown in Fig. 4.

The next step in Module 3 was to use the IBM Watson engine to train the ML model, which results in a ready-to-use server. Once the server was ready, the students constructed client software that interacted with the model and validated its prediction accuracy. The client software was created with Scratch. The corresponding screenshot is presented in Fig. 5.

As shown in Fig. 5, the egg character instructs the user to type in a sentence that the ML system classifies. Once the user types in the sentence, the egg character outputs the classification the ML model returned and its accuracy probability. During in-class observations, the students who created the example in Fig. 5 explained they added the saxophone just for laughs. The IBM-Watson engine does not support classification of text in Hebrew; hence Fig. 5 shows the student's project with text in English, including mistakes.

Module 4 — Constructing rule-driven ML system

In this module students moved from modifiers of ready-made ML systems, as was done in Module 2, to actual creators, but in a different way than in Module 3. In Module 4, they did not interact with a ready-made ANN, but instead constructed it themselves. The neural network algorithm that the students were asked to create was based on the logic gate activation function. To enable a Constructionist pedagogy, we used the Single-Neuron toolkit. As ML algorithms have become more complex, their reasoning and the rationales for their judgments have become less accessible and difficult to examine. This so-called "black box" problem has been acerbated by recent developments in deep learning that make use of very complex multi-layered neural networks (Webb et al., 2019). For this reason, to enable students to understand the ML algorithm we asked the students to create an ANN consisting of just one neuron.

Initially, they learned about logic gates through examples of an AND gate and an OR gate. Then they learned about truth tables. Next, they created an ANN to operate a simulation of a water system with two faucets that could be in only one of two modes, either open or shut. After programming it, the students tested it on a truth table of an AND gate, and then changed their inputs to an OR truth table and tested it again. They were encouraged to monitor the neural network's weight to better understand why/where the system was wrong in the initial training iterations and improve its accuracy as the training process integrated additional inputs. At the end of the module the students enhanced their ML system with an additional programmed character that explained the system. This gave them time to reflect on the system's purpose and mechanisms.

One student's project was a single neuron system that controlled the output of a water pipe flow according to the OR gate. The student set the truth table values to reflect the OR gate rules, and constructed the neuron model. The little green monster character in the system acted as the 'storyteller' who was programmed by the student to explain the system as shown in Fig. 6.

3.2. Method and procedure

To examine the students' acquisition and development of CT ML, the two different ML courses were administered to elementary school students, and addressed the following research

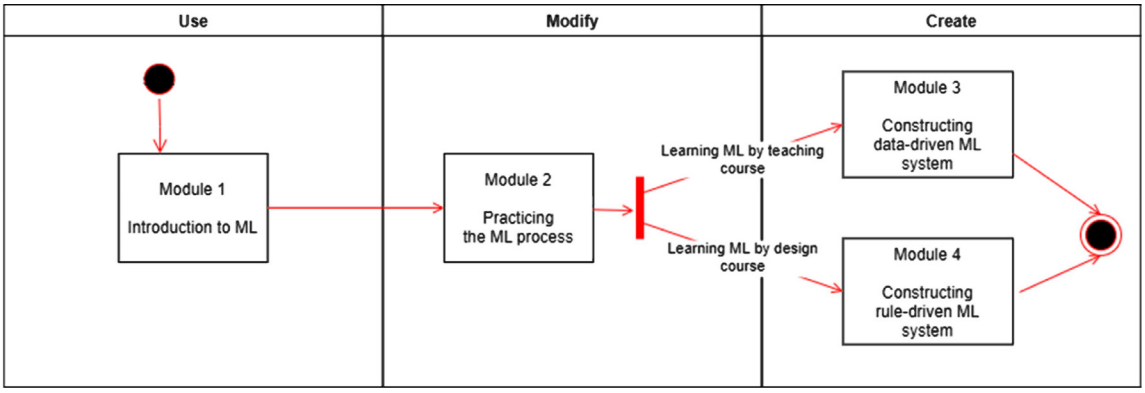

Fig. 3. A sequence diagram of the order of the modules for the two ML courses.

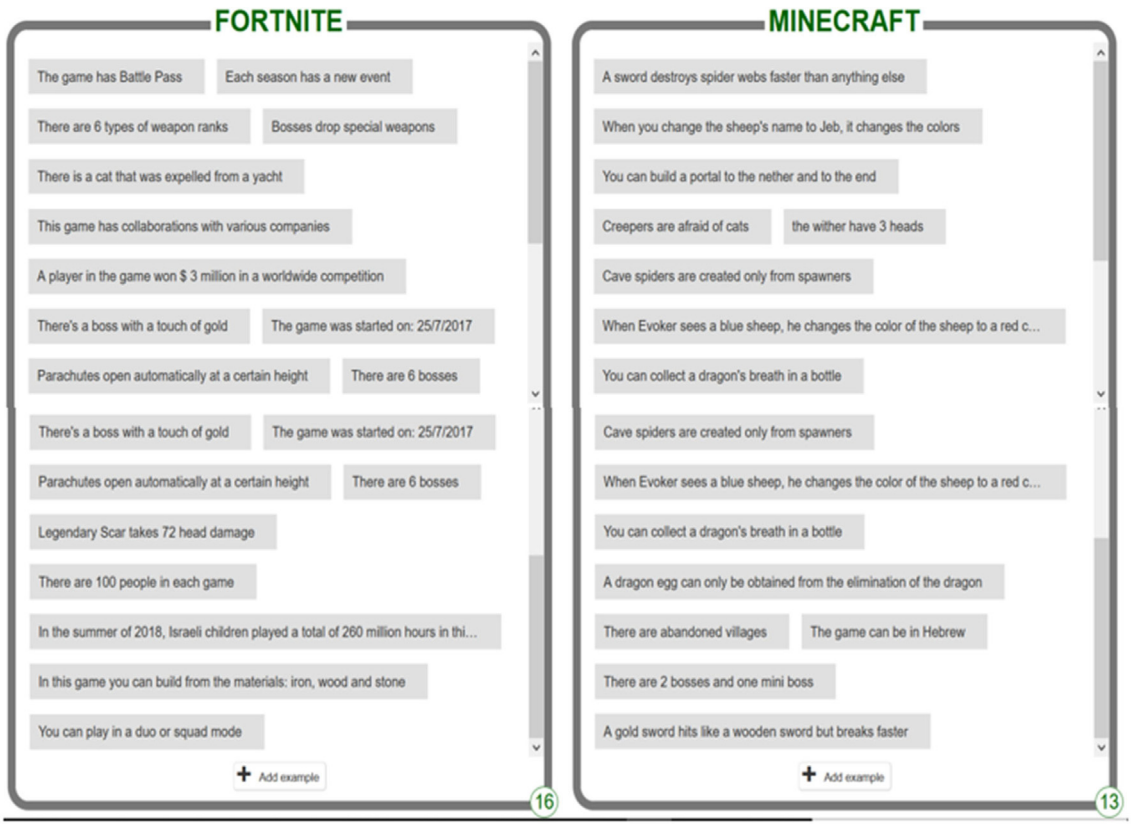

Fig. 4. A student's Fortnite/MineCraft training dataset in Module 3 of the course.

questions: In what ways are students' CT reflected in ML learning using the Constructionist approach? Do the 'Learning ML by Design' and 'Learning ML by Teaching' courses differ in effectiveness and if so, how?

This pilot study was conducted with two groups of students. Seven students participated in the Learning ML by Design course and eleven in the Learning ML by Teaching course. All the students were in 6th grade, were 12 years old, and volunteered for the 12-hour, teacher guided, online ML course. Each participant and their parents gave their informed consent to participate in the study.

This study evaluated the growth of students' CT competencies as they experienced a novel Constructionist ML course for elementary school children during the 2020 academic year. A sequential explanatory mixed-method approach was used to evaluate the quantitative and qualitative data (Shamir & Levin, 2020). Data collection and analysis were based on evaluations. A detailed description of each instrument is provided below.

Prior to the course, the students' basic acquaintance with online conference tools and programming with MIT Scratch were assessed. In Israel many students learn computer science (CS), mainly Scratch programming, starting in 4th grade. Participants in this study who did not have a CS background had to complete self-learning tasks based on the CS curriculum of the Ministry of Education prior to the intervention course. Students took a test at the start of the course. Each course module was composed of a set of activities and a variety of test activities. A typical lesson lasted two hours and students took a total of six lessons. After the course, a post-test was administered. Semi-structured interviews were also held with small groups which were transcribed for further analysis.

Students completed the evaluations individually, from their homes, using their computers during the online course. The tests

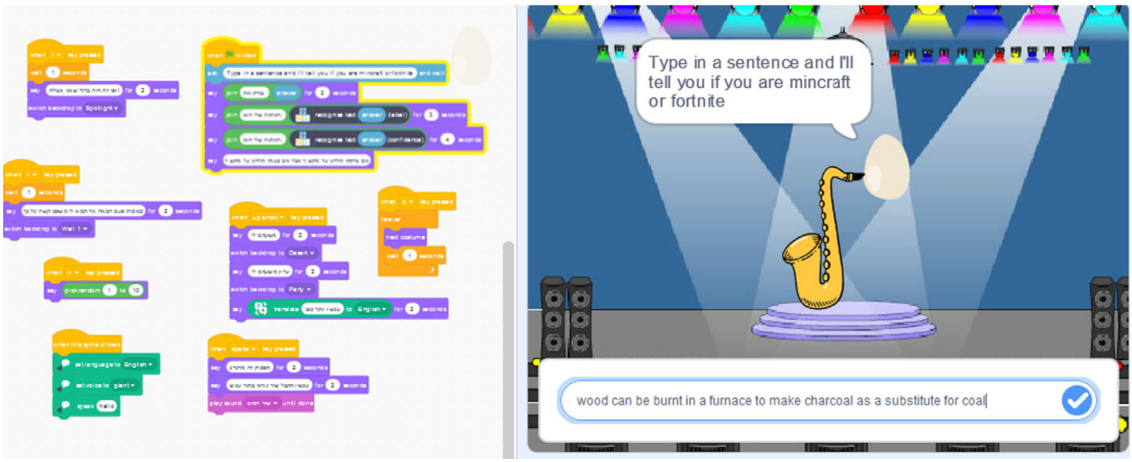

Fig. 5. Screenshot of client software created by a student.

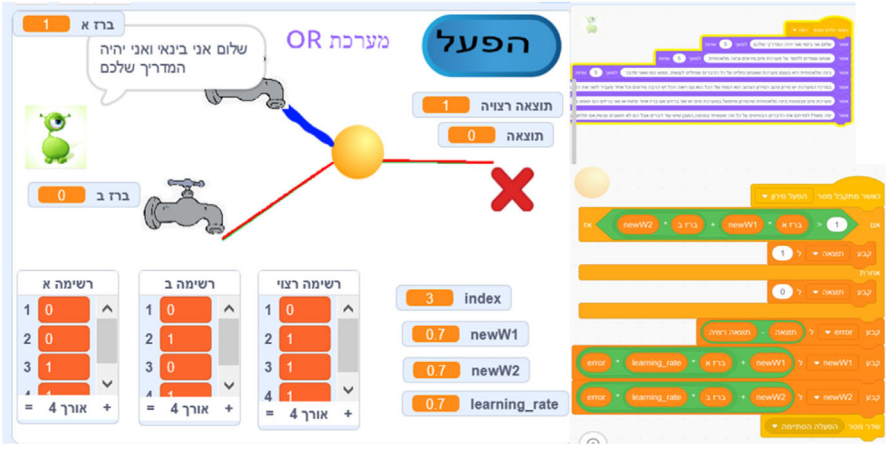

Fig. 6. Screenshot of OR gate neuron-based water system (captions are in Hebrew).

were made up of closed and open-ended questions on perspectives on ML, knowledge of ML processes, CT competencies, and interest in learning ML. The pretest consisted of 19 questions, and the posttest was comprised of 32 questions. The questions investigated enjoyment of course activities, self-efficacy with ML construction, and CT ML competencies in depth. The questions related to CT competencies were coded, since each question targeted a specific skill. The test items were written by the researchers specifically for this study. The qualitative validity ratings as evaluated by three expert judges indicated an Aiken V above 0.80 on all items.

To the best of our knowledge, ML learning self-efficacy has not been reported in the literature. For this reason, the questionnaire was based on a Constructionist validated robotics learning (Tsai, Wang, Wu, & Hsiao, 2021) questionnaire modified for ML construction. For example, instead of "I can make a robot", the item was phrased as "I can make a ML system". Other items were: "I can discuss how to make ML systems easily with peers" and "I can propose ideas for using ML to solve problems". The self-efficacy items were evaluated on a 5-point Likert scale, ranging from 1 (not confident at all) to 5 (very confident).

To examine students' motivation to learn we implemented the ARCS Model of instructional design (Keller, 1987) which is based on empirical investigations to assess students' motivation. The model is composed of four conceptual categories that subsume many of the specific concepts and variables that characterize human motivation.

1. Attention — should be sustained during the learning process.
2. Relevance — why this material is important to me.
3. Confidence — can influence a student's persistence and accomplishment.
4. Satisfaction — makes people feel good about their accomplishments.

To study the Feature selection skill, pre- and post-proficiency questionnaires were used. The questionnaires asked students to identify features in a dataset, and to create a mind map of features for a given dataset.

Students were asked to create a final project working alone or in pairs. Each of the two courses required a different type of project. In both, they programmed an ML system, but in one, they created their own ANN based on the Single-Neuron toolkit, while in the other course, they created an ML system based on machine-learning-for-kids toolkit, as described above.

Semi-structured interviews were conducted in small groups to give participants ample opportunity to express their thoughts in detail. Interviews were 30 min long.

The interview protocol was organized into several major sections:

1. Breaking the ice

a. Why did you decide to sign up for this course?

b. How was the course for you? (Students were referred to the shared bulletin board used during the course to refresh their memory)

2. About ML
a. How would you describe ML to friends or family?
b. How can a ML system help people?
c. Had you had the opportunity to interact with any type of ML system, what would you like it to be able to classify?
3. Final course project
a. How did you get started making your project?
b. What happened when you got stuck?

During the course, an online shared bulletin board tool was used to post ideas and discuss them. This was used to capture students' thoughts and discuss them in the final interviews.

4. Results

The fact that the thinking corresponding to ML differs significantly from more widely known computer thinking has attracted attention in the literature and has fueled numerous scientific debates (Denning & Tedre, 2021; Shapiro, Fiebrink, & Norvig, 2018). Therefore, the inclusion of ML in a CT course calls for answers to important questions about the relationship between ML and CT.

For this reason, Table 1 lists how ML CT practices relate to traditional CT practices. To date, there is scant CT ML research; thus, in this study we examined students' CT competency in terms of computational perspectives, machine training practices, and machine validation practices. This is the essence of the study. In addition, we used the qualitative methodology of grounded theory (Shapiro et al., 2018; Strauss, & Corbin, 1997) to characterize the effects of the ANN construction activity on the students.

The next section compares the students' CT gains relative to an ML course which did not involve the construction of an ANN (see Fig. 3 for a visual breakdown of the modules of the two courses). ANN is considered difficult to construct, therefore we specifically wanted to assess its effects on students' CT to better understand its role in a CT curriculum.

Students' CT competency was examined in terms of computational perspectives, machine training practices, and machine validation practices. The perspectives data from the pre/post questionnaires were rated on a Likert scale ranging from 1 to 5. The training and validation proficiency data were graded from 0 to 100. The results served to compare the Learning ML by Design course and the Learning ML by Teaching course as detailed below.

4.1. Computational perspectives

The computational perspective results are presented in Fig. 7.

Analysis of the post questionnaires in both courses indicated a 5% increase in motivation as compared to the beginning of the course. Given that the students chose to take the course, it was natural that their initial motivation was high, so an increase, even despite the fact that the course was challenging, was meaningful. This was further supported by the fact that during the course and in the final interview, participants asked to participate in a future course to learn more. The Learning ML by Teaching course participants scored an average of 4.28 out of 5 while the Learning ML by Design score was higher at 4.42. This may indicate that the learning activity of creating an ANN using the Single-Neuron tool kit had a greater impact on students' motivation to learn than interacting with a pre-made ANN using code.

Self-efficacy with modeling: Self-efficacy is an important part of successful learning. Thus, we examined modeling confidence using self-efficacy items. The comparison of the post-questionnaire Learning ML by Design course results to the Learning ML by Teaching course results indicated that the Learning ML by Teaching course participants scored an average of 3.2 out of 5. On the other hand, the Learning ML by Design score was 21% higher at 4.28. This may indicate that constructing an ANN hands-on contributed to students' sense of capability.

Understanding ML processes: Using a questionnaire and interviews, we collected data on what the participants considered key to good ML system training and evaluation. For example, during the course they were shown a video of an autonomous car that failed to stop at a stop sign. In the interview one student explained this error: "The car needs to be trained better. If you want it to identify these things, you put the sign in all sorts of places, not just in one place, so the car can be trained in different situations to recognize the sign".

The Learning ML by Teaching course post-questionnaire had an average score of 4.6 which was lower than the Learning ML by Design course score of 5 out of 5. This may indicate that constructing an ANN can enhance students' ML understanding.

Overall, the scores for the computational perspectives on the Learning ML by Design course were higher, thus confirming our hypothesis that constructing an ANN using appropriate scaffolds such as the Single-Neuron toolkit contributes to young students' computational thinking.

Students were asked in the interview what ways they thought a ML system could help people. Their answers varied in content: *"I would like to have a ML robot who is a soccer referee which sees everything like if someone touches the ball by hand or is offside. It would be better if the robot was not on the field because it interrupts the players"*. Another student suggested using ML to replace judges in a court of law. Yet another student suggested that ML could help the blind by calling out red lights and other features of the environment it was trained to identify. Another suggested an autonomous scooter, so parents will not need to drive students to after school activities.

4.2. Machine training practices

The results of CT practices related to ML training are presented in Fig. 8.

Data split proficiency: The end of course evaluation aimed to assess the students' ability to split a set of images into two groups: one for training and validation. The purpose was to determine whether the students would reason in terms of 3 dimensions: (1) using as many of the full set of images as possible (2) not using the same image for both datasets (3) splitting each pair of images with common features in the two datasets. In both courses (see Fig. 8) students scored high, exceeding 0.75. In addition, the students had a good grasp of data diversity in terms of covering as many features as possible. However, they did not fully understand that a properly validated ML machine should not be tested with the same images as used during training.

Data filtering proficiency: This part of the evaluation assessed the students' ability to filter out data before the training phase. For instance, when creating an ML system to classify images into a "Zebra" category and a "Pedestrian crossing" category, having images with both zebra and pedestrian crossing may cause the machine to be trained incorrectly. In one of the questions that tested data filtering proficiency, the students were given a set of images (see Fig. 9). They were asked to decide which images to remove from the dataset. This test was designed to determine whether the students would reason in terms of the following 2 dimensions: (1) Remove images with both a zebra and a pedestrian crossing; (2) Filter out images of a pedestrian crossing that was painted to look like a zebra. The findings indicated that students did better on dimension 1 than dimension 2 but that the scores in general were low (score $\leq$ 0.35, see Fig. 8).

Category Selection Proficiency: Category selection skills make it possible to find similarities, including perceptual distances, with

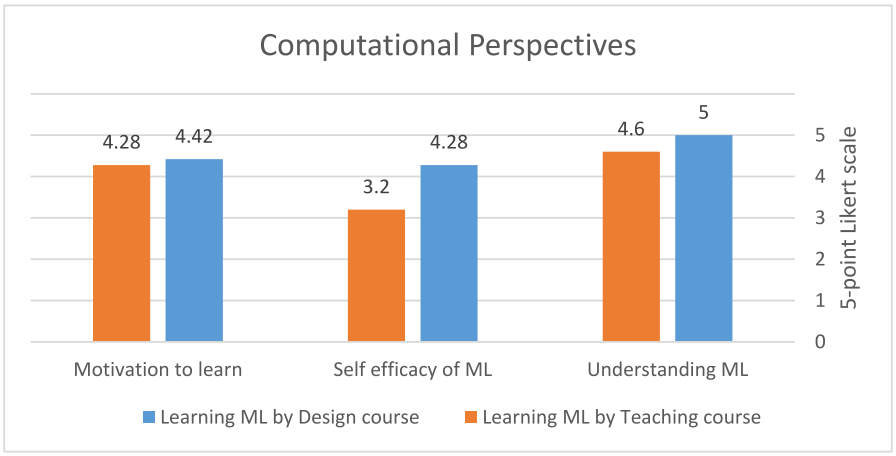

Fig. 7. Results of computational perspectives for the Learning ML by Design course vs. the Learning ML by teaching course.

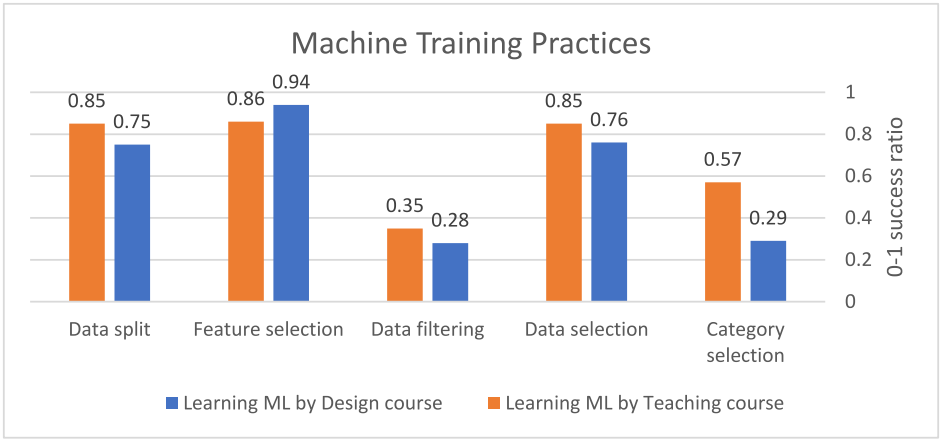

Fig. 8. Results of machine training practices per course.

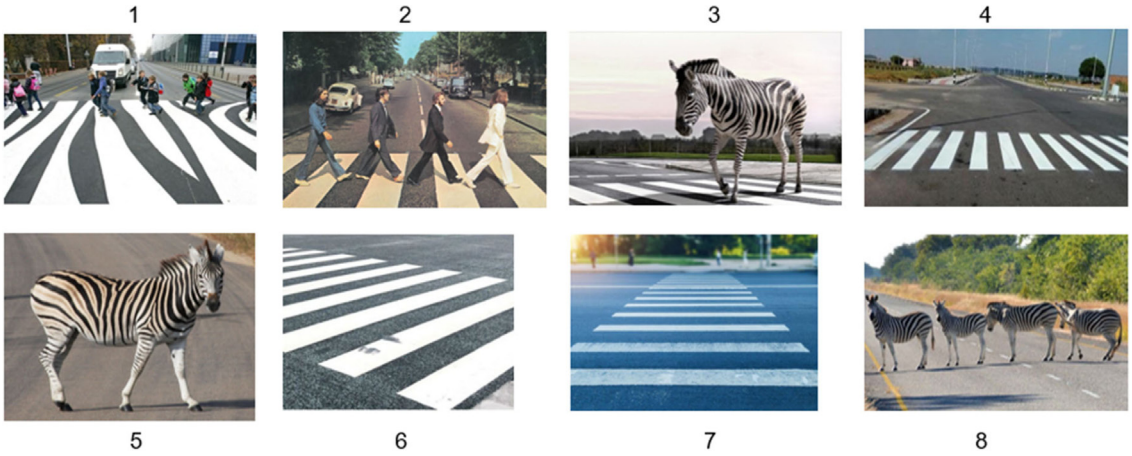

Fig. 9. The dataset of images to be filtered by the students.

prototype examples of categories. The rationale stems from human perception studies which posit that these categories are not defined by lists of features but rather by similarity to prototypes. The analysis here implemented this argument that categories are not merely a list of features. The evaluation showed that the students did less well when asked to find a category (score $\leq$ 0.57, see Fig. 8) than when asked to find features for a pre-defined category (score $\geq$ 0.76, see Fig. 8). In-class observations showed that students frequently switched categories on a task that asked them to select a category for classifying fish images out of a closed set of obfuscate categories such as "Glitchy" or "Awesome" (see Fig. 10). The students were asked in class to classify images as "true"; i.e., fit the category, or "false"; i.e., did not fit the category. The students verbally explained that it was too difficult for them to justify some of the categories, which is why they switched them during the task.

An interesting observation emerged from an activity that asked the students to create their own categories. One student

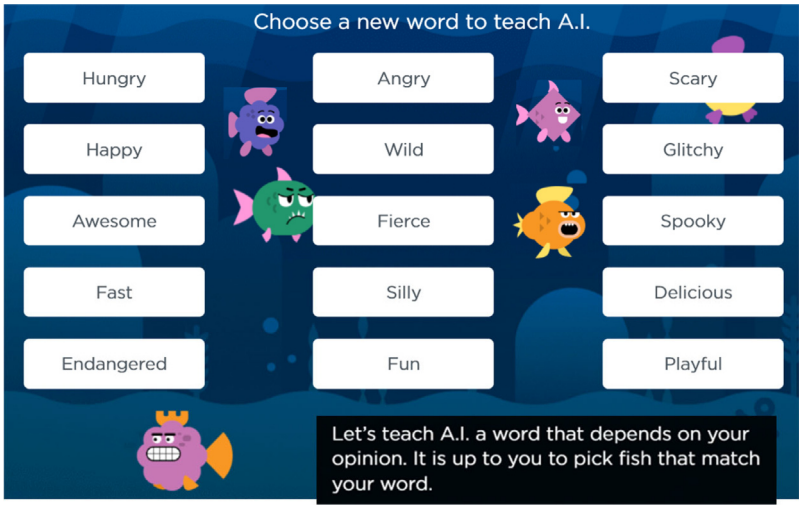

Fig. 10. The category selection task.

came up with algorithm primitives (e.g., words which are part of the algorithm lexicon) as categories. He wanted to categorize sentences into two types of games: Fortnite or Minecraft. The five categories he created were "If", "Input sentence", "Equals", "Fortnite", "Minecraft". The latter two categories represent proper category selection, but the former three are incorrect.

Data selection proficiency: Data selection is the ability to select data that are most likely to improve the system's prediction ability and avoid bias. This skill requires considering the different features of the object to be labeled and selecting a diverse dataset. For instance, when creating an ML system to identify images with sheep, insufficient data selection occurs when all the sheep images have greenery in the background. This could cause the ML system to mistakenly categorize an image as having sheep even if it only shows a green pasture. In the interviews we noticed that the students did not consider that sheep with only a pasture in the background was a problem. On the other hand, when evaluating data selection skills on a multiple-choice test, the students scored relatively high (score $\geq$ 0.76, see Fig. 8).

Feature selection proficiency: Feature selection is the process of identifying attributes that contain relevant information for an ML system's data classification. The students were evaluated by tests, tasks and interviews, and during class observations. We created several assessment tools to analyze feature selection, including the number of features and their validity ratio. Pilot studies showed that these skills are extremely challenging; therefore, a great deal of care went into the learning module, including a specific task of creating a feature map using explicit instructions on decomposition skills. The students scored relatively high (score $>$ 0.86, see Fig. 8) on this skill.

A comparison of the scores in the two courses showed that the differences in proficiency were small (difference $<$10%) except for the Category Selection skill. In this case, Learning ML by Teaching course participants scored an average of 0.57 while the Learning AI by Design score was 18% lower at 0.29. This may suggest that the enhanced engagement with category selection had a stronger influence on this skill than creating an ANN.

4.3. Machine validation practices

The results of CT practices related to ML validation are presented in Fig. 11.

Prediction proficiency: Prediction refers to forecasting the likelihood of an ML model's outcome after it has been trained on a historical dataset and applied to new data. The post-test included a set of questions that evaluated the ability to predict. It presented a trained dataset of labeled images and asked the students to predict the result of a new image and to explain why. For instance, the students were presented with a training dataset of 6 images the ML system had used to learn whether an object "belongs in the ocean", labeled by "YES", or "does not belong in the ocean", labeled by "NO". The 7th image required a prediction (see Fig. 12). On this specific question, most participants predicted the machine would classify the image of a blue can as "belongs in the ocean". This indicates that they were able to put aside the human intuition of predicting that a can "does not belong in the ocean" but instead invoked their ML CT skills. Most student explanations referred to the "color feature". In other words, the four objects tagged as "yes" (belongs in the ocean), were blue, as was the can. Thus, it was more likely that the ML system would predict that the can belongs in the ocean. They explicitly used the word "features" in their justification, strengthening our claim that they had assimilated the notion of feature selection. The results showed that most students scored well (score $>$ 0.93, see Fig. 11) on this skill.

One of the students who gave a wrong answer explained it was based on an "eyes" feature, saying, "The image of the can object does not have eyes and neither do the two objects that were labeled as do not belong in the ocean". This reasoning accurately takes the feature into account but fails to acknowledge that eyeless jellyfish was also labeled as "Yes",. This response was thus scored as incorrect for having chosen 'eyes' as a differentiating feature.

Evaluation: Evaluation is the skill related to assessing how well the ML system categorized the validation dataset and how the results could be improved so that the labeling could be enhanced. Possible issues involved in evaluation included whether the machine was under-trained, whether the model was biased towards a certain object, whether the features were appropriately selected and whether the dataset was split appropriately. One of the multiple-choice questions to test evaluation skills on the post-test was: "The ML system was trained to identify pictures of "food" or "not food". When tested, it gave inconsistent results for sandwiches, often putting them in the "not food" class. What can explain this?": (a) The person who trained the system does not like sandwiches and did not put pictures of them into the training set; (b) sandwiches are not food, so the system is wrong; (c) the system is correct about oranges, so there is no problem with it; (d) the programmer developed the system with a bug. The correct

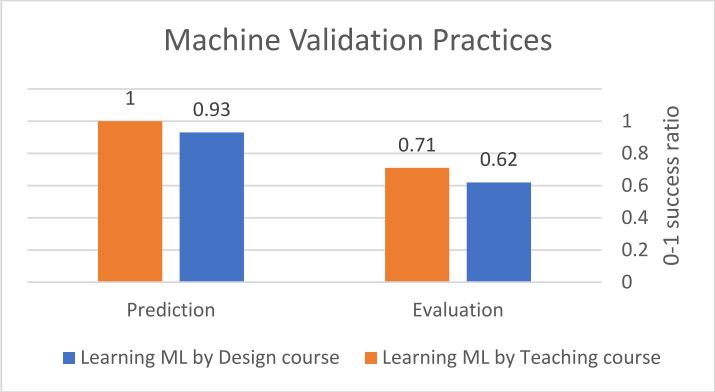

Fig. 11. Results of machine validation practices for each course.

Fig. 12. Prediction skill question . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

answer is 'a'. When comparing the course results, the Learning ML by Teaching course students scored an average of 0.71, whereas the Learning ML by Design score was lower at 0.63. This may imply that enhanced engagement with data throughout the ML process had a stronger influence on the student's evaluation skills than creating an ANN which only had two Boolean inputs: 0 or 1.

5. Discussion, limitations and future work

Today, people are surrounded by machine leaning (ML) based technology therefore it is only natural that learning ML will is finding its way into education. The inclusion of ML in a computational thinking (CT) course requires answers to fundamental questions about the relationship between ML and CT: Is ML part of CT or an extension to it? Does our concept of traditional CT change after integrating ML components?

This article presents findings exploring the gains of CT competencies of elementary school students who took part in two different ML courses that implemented constructing ML systems using a Constructionist pedagogy.

The results showed that the students learned to successfully create a computerized ML system and train it to classify input datasets. They showed knowledge of how data should be selected and filtered for the system to learn with high probability of making a successful prediction. They acquired ML concepts such as datasets, features and ML-bias, as well as the difference between data-driven and rule-driven programming. They were able to develop rich ML projects using their own categories, datasets and evaluations. It was clear that students came up with interesting, diverse ML systems for expressing themselves through the constructionist activities.

Several ML environments were used by the students in the two constructionist courses. The Learning ML by Design course involved students in programming their own ANN, thus giving them a white box view of a ML system. The Learning ML by Teaching course involved using a preexisting ANN, thus giving them the ANN as a black box without understanding its internals.

Both courses had favorable results with regards to students gains but there were some differences between the outcomes. The participants exhibited slightly greater computational perspectives in the Learning ML by Design course than in the Learning ML by Teaching course. This may point to the advantages of teaching a Constructionist ANN activity in a ML course. It may indicate that the worldview the students formed about their surroundings and about themselves was enhanced when they worked on a ML algorithm that enabled the machine to learn.

On the other hand, students performed better in the Learning ML by Teaching course with respect to computational practices. The course had more data-driven activities than the Learning ML by Design course. The skills that resulted in high proficiency in the Learning ML by Teaching course were category selection, data selection, data filtering, data split, category prediction and result evaluation. These are all practices that are profoundly data-driven, which may explain the students' greater gains in CT practices since these skills are based on the identification of data attributes.

Research on the educational possibilities afforded by ML construction in elementary school is in its infancy. This study provides new findings and pedagogical insights for future research, development, and educational efforts. It suggests key design considerations for future development of a Constructionist approach to ML learning. It presents teacher-guided activities that can be implemented in school. The activities are adapted for elementary school students' mathematical background and incorporates the appropriate scaffolds. The modules are scalable so students can be creative with their own intelligent machines and put forward

models to address a wide variety of problems. This work can be extended to different age groups and more ML platforms can be added to the curricula. This study thus paves the way for incorporating a Constructionist ML learning approach into the elementary school CT curriculum. It addresses the need for constructionist approaches to teaching ML. A CT-ML framework was introduced (Fig. 2) to set the foundations of assessing students' CT engaging in ML activities.

Nevertheless, this study has several limitations. It was conducted on a small group of students, all of whom were volunteers. While such small-scale qualitative studies are useful for in-depth exploration of a phenomenon, they do not allow for generalization beyond the sample under investigation (Ivankova, Creswell, & Stick, 2006). In future research, the results should be verified on a larger sample. Since the course was online and not in a regular class mode, in-person classes might yield different results.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Declaration of funding source

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Selection and participation of children

This study was approved by our institution's relevant ethics committee. We recruited elementary school students through their parents. We sent the consent form to fully inform parents and youth prior to signing up. Each participant and their parents gave their informed consent to participate in the study. The consent forms for participation were signed and obtained from parents prior to the study. The children were informed that they could withdraw from participation at any given time.

References

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference. *Future of Learning Group Publication*, *5*(3), 438.

Badie, F. (2016). Concept representation analysis in the context of human-machine interactions. In *14th international conference on e-society (ES 2016)* (pp. 55–62). International Association for Development, IADIS.

Blikstein, P., & Wilensky, U. (2009). An atom is known by the company it keeps: A constructionist learning environment for materials science using agent-based modeling. *International Journal of Computers for Mathematical Learning*, *14*(2), 81–119.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada, Vol. 1* (p. 25).

De Vries, M. J. (2008). Gilbert simondon and the dual nature of technical artifacts. *Techné: Research in Philosophy and Technology*, *12*(1), 23–35.

Denning, P. J., & Tedre, M. (2021). Computational thinking: A disciplinary perspective. *Informatics in Education*.

Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

Druga, S., Vu, S. T., Likhith, E., & Qiu, T. (2019). Inclusive AI literacy for kids around the world. In *Proceedings of FabLearn 2019* (pp. 104–111).

Grover, S. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In *Emerging research, practice, and policy on computational thinking* (pp. 269–288). Cham: Springer.

Guyon, I., & Elisseeff, A. (2006). An introduction to feature extraction. In *Feature extraction* (pp. 1–25). Berlin, Heidelberg: Springer.

International Society for Technology in Education & Computer Science Teachers Association (2011). Operational definition of computational thinking for K–12 education. Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf.

Ivankova, N. V., Creswell, J. W., & Stick, S. L. (2006). Using mixed-methods sequential explanatory design: From theory to practice. *Field Methods*, *18*(1), 3–20.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *Future directions in computer science education summit meeting, Orlando, FL*.

Kahn, K., & Winters, N. (2020). Constructionism and AI: A history and possible futures.

Keller, J. M. (1987). Development and use of the ARCS model of instructional design. *Journal of Instructional Development*, *10*(3), 2–10.

Khalid, S., Khalil, T., & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference* (pp. 372–378). IEEE.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson …, J., & Werner, L. (2011). Computational thinking for youth in practice. *Acm Inroads*, *2*(1), 32–37.

Leiser, D. (1996). Constructivism, epistemology and information processing. *Anuario de Psicología/The UB Journal of Psychology*, (69), 93–114.

Levin, I., & Mamlok, D. (2021). Culture and society in the digital age. *Information*, *12*, 68, 2021.

Levin, I., & Tsybulsky, D. (2017). The constructionist learning approach in the digital age. *Creative Education*, *8*(15), 2463.

Mariescu-Istodor, R., & Jormanainen, I. (2019). Machine learning for high school students. In *Proceedings of the 19th Koli calling international conference on computing education research* (pp. 1–9).

McCarthy, J., Minsky, M. L., Rochester, N., & Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, *27*(4), 12.

Ministry of Education Israel (2021). Computational thinking. Retrieved May 1, 2021, from https://pop.education.gov.il/teaching-practices/search-teaching-practices/computational-thinking/.

Papert, S. (1980). *Mindstorms: Computers, children, and powerful ideas* (p. 255). NY: Basic Books.

Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, *36*(2), 1–11.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan ., K., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.

Rouhiainen, L. (2018). *Artificial intelligence: 101 things you must know today about our future*. Lasse Rouhiainen.

Shamir, G., & Levin, I. (2020). Transformations of computational thinking practices in elementary school on the base of artificial intelligence technologies. In *Proceedings of EDULEARN20 conference, Vol. 6* (p. 7th).

Shamir, G., & Levin, I. (2021). Neural network construction practices in elementary school. *KI-Künstliche Intelligenz*, 1–9.

Shapiro, R. B., Fiebrink, R., & Norvig, P. (2018). How machine learning impacts the undergraduate computing curriculum. *Communications of the ACM*, *61*(11), 27–29.

Strauss, A., & Corbin, J. M. (1997). *Grounded theory in practice*. Sage.

Tedre, M., & Denning, P. J. (2016). The long quest for computational thinking. In *Proceedings of the 16th Koli calling international conference on computing education research* (pp. 120–129).

Tsai, M. J., Wang, C. Y., Wu, A. H., & Hsiao, C. Y. (2021). The development and validation of the robotics learning self-efficacy scale (RLSES). *Journal of Educational Computing Research*, Article 0735633121992594.

Vartiainen, H., Tedre, M., & Valtonen, T. (2020). Learning machine learning with very young children: Who is teaching whom? *International Journal of Child-Computer Interaction*, *25*, Article 100182.

Webb, M., Fluck, A., Deschenes, M., Kheirallah, S., Lee, I., Magenheim ., J., & Zagami, J. (2019). Thematic working group 4-state of the art in thinking about machine learning: Implications for education.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Yegnanarayana, B. (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd..

Zimmermann-Niefield, A., Turner, M., Murphy, B., Kane, S. K., & Shapiro, R. B. (2019). Youth learning machine learning through building models of athletic moves. In *Proceedings of the 18th ACM international conference on interaction design and children* (pp. 121–132).