Received: 24 July 2020 | Accepted: 8 March 2021

DOI: 10.1111/bjet.13088

ORIGINAL MANUSCRIPT

British Journal of Educational Technology BERA

Constructionism and AI: A history and possible futures

Ken Kahn | **Niall Winters**

Department of Education, University of Oxford, Oxford, UK

Correspondence
Ken Kahn, Department of Education, University of Oxford, 15 Norham Gardens, Oxford OX2 6PY, UK.
Email: toontalk@gmail.com

Funding information
None

Abstract

Constructionism, long before it had a name, was intimately tied to the field of Artificial Intelligence. Soon after the birth of Logo at BBN, Seymour Papert set up the Logo Group as part of the MIT AI Lab. Logo was based upon Lisp, the first prominent AI programming language. Many early Logo activities involved natural language processing, robotics, artificial game players, and generating poetry, art, and music. In the 1970s researchers explored enhancements to Logo to support AI programming by children. In the 1980s the Prolog community, inspired by Logo's successes, began exploring how to adapt logic programming for use by school children. While there have been over 40 years of active AI research in creating intelligent tutoring systems, there was little AI-flavoured constructionism after the 1980s until about 2017 when suddenly a great deal of activity started. Amongst those activities were attempts to enhance Scratch, Snap!, and MIT App Inventor with new blocks for speech synthesis, speech recognition, image recognition, and the use of pre-trained deep learning models. The Snap! enhancements also include support for word embeddings, as well as blocks to enable learners to create, train, and use deep neural networks. Student and teacher project-oriented resources

highlighting these new AI programming components appeared at the same time. In this paper, we review this history, providing a unique perspective on AI developments—both social and technical—from a constructionist perspective. Reflecting on these, we close with speculations about possible futures for AI and constructionism.

KEYWORDS
artificial intelligence, constructionism, deep neural networks, logic programming, Logo, machine learning, project-based learning, Snap!, TensorFlow.js

Practitioner notes

What is already known about this topic
- There exist excellent broad surveys of the current status of teaching machine learning in schools, for example Marques et al. (2020). There are historical collections of AI and education research papers that include descriptions of constructionist activities, for example Yazdani (1984).

What this paper adds
- This paper adds an in-depth focus on historical and current efforts on AI and education that support constructionist teaching. This focus enables us to delve deeper than a broad survey. Uniquely, we provide a 50-year historical perspective on constructionist AI tools, trials, and research. Grounded in this history and our survey of current tools and projects, we provide speculations about future directions.

Implications for practice and/or policy
- We hope our descriptions of current AI programming tools for non-experts placed in a broad historical context will be of use to teachers wishing to introduce AI to their students in a constructionist manner, as well as to developers and researchers aiming to support such teaching.

INTRODUCTION

A key question asked by this special section is, as constructionism moves forward, what does it have to offer both within education and wider society? In this paper, we address this question by focusing directly on the relationship between constructionism and artificial intelligence (AI).

There is no doubt that artificial intelligence is impacting the educational landscape in fundamental ways (Luckin et al., 2016). In this paper, we focus on the long history of the relationship between AI and constructionism. However, a confluence of factors may mean that new AI algorithms will promote the aims of those with power, furthering the exclusion of marginalised learners. This we have termed digital structural violence (Winters et al., 2020) to draw attention to ethical issues related to AI in education. One way in which this potential future can be tackled now is to expose young people and teachers to the positive as well as negative possibilities of AI as early as possible through constructionist approaches in education. We do this within the frame of an emancipatory aesthetics of education (Biesta,

2020), where the key aim is to 'bring something into the student's "field" of perception, which can be visual, auditory, sensory and perhaps may even include the touching of the student's soul' (ibid.), in this case, by constructionist microworlds that challenge them, and allow for the teacher to help them move outside of themselves.

This approach harks back to Constructionism's earliest days when AI's genesis in education was thought about as a force for good, particularly in the fields of children's programming (Kahn, 1977). Reflecting on this time for a moment, it becomes clear constructionism and artificial intelligence were intimately entwined. In these early days, the idea that artificial intelligence could model student cognition, thus offering ways of understanding the mind, was prevalent. In concrete terms, there was a clear relationship between constructionism and symbolic artificial intelligence. More recently, this has changed with the wider shift in AI towards machine learning with neural networks and big data, which constructionism has mirrored.

AI was there from the beginning

In 1966 Seymour Papert consulted for Wally Feurzeig, head of the education group at BBN. Papert decided that what was needed was a programming language specifically designed for children. During that academic year, Papert talked about this new language mostly with Danny Bobrow (who had recently completed his doctoral thesis at the MIT AI Lab and was now head of BBN's AI Group) and Feurzeig. By the end of summer 1966, Papert specified this new language and presented it to the small BBN group. The group saw Logo as a 'baby' version of the Lisp AI language. Bobrow immediately started implementing the language in Lisp and then passed it along to Cynthia Solomon and Dick Grant. In 1969, Papert and Solomon left BBN and formed the Logo Group at the MIT AI Lab. (Solomon et al., 2020).

The core idea of constructionism is that learning 'happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity' (Papert & Harel, 1991). To 'consciously engage' one needs to bring to bear previously acquired concepts while one is introspecting. Within constructionism it is well known that computational concepts are an important part of this. Here, however, we focus, not on computation per se, but on the learner's ideas about how they learn, represent things, solve problems, and create. AI-oriented constructionist projects may help learners acquire deeper and more effective ways of learning and creating. These constructionist ideas (long before they were labelled 'constructionist') influenced the design of the Logo language (Solomon et al., 2020) and early Logo project ideas. The paper 'Twenty Things to Do with a Computer' (Papert & Solomon, 1971) suggests on the first page that artificial intelligence should deeply affect our thinking about 'computers in education'. Several of the 20 project suggestions were AI projects (as it was conceived in the 1970s) such as programming robots with sensors and effectors, composing music, and generating poetry. These were soon followed by projects such as generating grammatical sentences and programming AIs to play simple games like *Tic Tac Toe* and *Nim*.

In 'Three Interactions between AI and Education' (Kahn, 1977) AI plays three roles: (1) students use AI programming tools in their projects, (2) students create artefacts by interacting with AI programmes, and (3) the use of computational theories of intelligence and learning in the design of learning activities. In the 1970s research prototypes were built upon Logo to support student projects involving natural languages and semantic networks. LLogo (Goldstein et al., 1975), an implementation of Logo in Lisp, was particularly well-suited for this.

While a focus on programming tools is necessary, it is important that their implementation is based on a strong underpinning rationale as to why children should create and interact with AI programmes (Kahn, 1977):

> ‘Children are encouraged to think explicitly about how they solve problems. Hopefully the children will thereby improve their ability to describe and understand their own thoughts’.
>
> ‘The problem domain to which the AI programs are applied is learned, and in a new and perhaps better way’.
>
> ‘If children are to program, then AI can an interesting open ended problem domain for that programming’.
>
> ‘The children will learn about AI which is a subject … that is as important as spelling or history’.

To this list we now add a fifth important point reflecting the role of AI in society today:

> To engage with the social implications of AI through building and critiquing AI training models.

Regarding the first point consider the role self-reflection plays in learning according to Marvin Minsky (2009, 2019) (original emphasis):

> An adequate theory of learning should also cover the ‘reflective skills’ that people use to recognise exceptions to generalisations, to eliminate tactics that waste too much time, and more generally, to make longer-range plans and form broader perspectives.
>
> But I am convinced that these ‘self-reflective’ processes [the methods that people use for *thinking about what we've been thinking about*] are the principal ones that people use for *developing new ways to think*.

During the 1970s Kahn built two microworlds in Logo to support student AI projects (Kahn, 1975, 1976). One effort involved supporting the building of projects that could produce simple animations in response to textual instructions. Pattern matching was used to extract the ‘meaning’ from the instructions. The style of animation was inspired by Minsky's Society of Mind Theory (Minsky, 1988). The other microworld provided higher-level support for natural language processing. While much simpler than later natural language toolkits, these efforts were designed to encourage students to think about language and how we generate sentences and extract meanings.

Another effort in that period to combine AI and constructionism was the work of Mark Miller on an intelligent programming tutor for Logo (Miller, 1979). Note that the goal here was to use AI to support students’ constructionist generic programming activities. In contrast, this paper focusses on efforts to support students in constructing AI projects.

Efforts in the 1980s to bring AI programming into schools

The efforts to bring AI programming into schools in the 1980s happened within a context of a boom in Logo's popularity. It is important to note here that constructionist approaches do not assume a ‘minimally invasive’ form of education, where the teacher is excluded (Noss & Hoyles, 1996).

The articles in the book *New Horizons in Educational Computing* (Yazdani, 1984) in the Ellis Horwood Series on Artificial Intelligence describe many efforts to make AI programming tools accessible to children. The logic programming language Prolog inspired two very different approaches: (1) providing children with a simple syntax for using a powerful deductive engine and (2) giving children a powerful symbolic programming language that provides support for building semantic networks and doing backtracking searches. Prolog programmes have declarative and procedural readings and different research groups focussed on each of these different aspects.

Bob Kowalski first introduced problem solving in Prolog to 12-year-old students at his daughter's school in 1978 (Kowalski, 2015). (He describes how they needed to connect to a university computer to run Prolog using a pay phone. He wrote 'The connection would be lost whenever our coins ran out'.) His paper 'Logic as a computer language for children' (Kowalski, 1984) describes the declarative approach. Students can build deductive, typically symbolic, databases. Ten-year-old children constructed semantic networks using a front end to Prolog with a very simple syntax. Sergot instead built a meta-interpreter for Prolog for use by children (Sergot, 1984). While much slower, it was able to provide explanations for any deductions it made. (Nichol & Dean, 1984) describes how students could use Prolog to explore history.

One issue that arose with these efforts is that while Prolog and similar languages have both declarative and procedural readings, the declarative reading was imperfect. In particular, it was not difficult to create a logically correct program that would not terminate. In recent times this has been addressed by systems that memorise previous goals. Answer Set Programming is an example that has been introduced to middle school students (Zhang et al., 2019).

Some of these efforts to introduce logic to students via programming continue to this day. These efforts are focused on *logic* and not artificial intelligence. But the distinction is far from clear since logic is well-suited for knowledge representation, problem solving, and question answering. An example is a research by Texas Tech University researchers who introduced Answer Set Programming to middle school students enabling them to create purely declarative models of food webs, atoms, and relative motion. These models were capable of answering a range of queries. While such logic programming systems can be used by students in a constructionist manner, we are not aware of any such uses.

There are many projects that can be supported by an intelligent data (or knowledge) base that some students will find personally meaningful. For example, a question answering system could be used by students to model a favourite topic such as sports statistics, a historical period, pop culture, etc.

An example of using Prolog as a powerful procedural tool is described in Kahn (1984). A grammar kit was built upon LM-Prolog that enabled students to build grammars to both parse and generate sentences. It relied upon a 'backtracking turtle' that erased its trail when backtracking. This visualisation helped students understand how their programmes and queries worked and to find and fix bugs.

POP-11 (Sloman, 1984) is an AI language similar to Lisp that supports semantic databases. Development of POP-11 started in 1974 and was heavily influenced by the Logo efforts at MIT and the University of Edinburgh. The goals of POP-11 were to enable students to, instead of 'making toy cranes or toy aeroplanes, or dolls', to make 'toy minds'. The POP-11 research team wanted 'people to experience themselves as computation'. Accessible AI programming tools were developed with these goals in mind.

POP-11 later incorporated Prolog into their system (renaming it Poplog) after seeing the successes others were having introducing Prolog to children. From 1983 to 1998 Poplog was an expensive commercial product but after it became free in 1998 it was once again

used by school students for AI programming. Examples of Poplog projects include puzzle solving, chatbots, analogical reasoning, and planning (Sloman, 2012).

Everything changes in 2017

The AI research community, the general public, and educators view of AI began changing in 2012 as neural networks began attaining impressive performance in image recognition, machine translation, transcription of speech, game playing (Atari games, Go, and chess), and natural language processing. Deep neural networks are what most people think of today when they hear AI. Deep learning combined with big data and special hardware for doing tensor operations now dominates the field.

2017 saw the appearance of several excellent machine learning tools and resources for children. They include:

1. April 2017 Dale Lane created *The Machine Learning for Kids* website. Learners can create neural networks for images, text, or numbers, train them, and use them in Scratch programmes. The site includes a long list of well-designed project suggestions (Lane, 2020).
2. May 2017 Stephen Wolfram added a machine learning chapter to the *Mathematica* online programming textbook. He published a blog describing the AI capabilities of the Wolfram Language. He suggests a range of projects suitable for middle school students (Wolfram, 2017).
3. May 2017 Google announced its first AIY project where students can build standalone artefacts capable of speech synthesis and recognition. This was followed by a similar kit for image recognition (Google, 2020).
4. September 2017 Kahn and Winters released the eCraft2Learn project's Snap! blocks that access web services for speech synthesis, speech recognition, and image recognition (Kahn & Winters, 2017). This was followed by Snap! blocks for using pre-trained TensorFlow models, and word embeddings, (Kahn & Winters, 2018). Subsequently, blocks for creating, training, and using neural nets were added a year later (Kahn et al., 2020).
5. October 2017 Google released its Teachable Machine which is a web page where one can train a model to classify images, sounds, or poses (Google, 2017). Unlike the other 2017 machine learning tools, this one did not have a programmer interface. While it is a good introduction to an aspect of machine learning it lacked the open-ended nature of most constructionist tools. It did, however, directly inspire similar projects in Snap! using the eCraft2Learn blocks. The significantly improved Teachable Machine 2.0 was released in November 2019 (Google, 2019). It provides support for saving and using student trained models in their JavaScript or Python projects.
6. During 2017 and 2018 Stefania Druga at the MIT Media Lab led the Cognimates Project which like the Machine Learning for Kids website added AI capabilities to Scratch (Druga, 2018). However, it built on this in new and interesting ways. The Cognimates platform, added modular extensions for cognitive services, in AI training capabilities (much like eCraft2Learn), in addition to current block features of coding apps such as Scratch. Several trials were conducted that demonstrated the students' increase in understanding of the strengths and weaknesses of AI.

This explosion of activity has continued. AI programming by children projects has since started at many universities.

Are AI curriculum efforts constructionist? The AI4K12 example

AI4K12 is a wide collaboration of researchers and educators focused on providing guidance for the teaching of AI in schools (Touretzky, 2020; Touretzky et al., 2019). They have organised several international workshops, created guidance for teachers, and curated a substantial collection of resources. They have organised their efforts around five big ideas as illustrated in the poster in Figure 1.

In light of the motivation for learning about AI as outlined in the introduction, the focus on 'societal impact' is critically important. AI increasingly plays a role in people's lives and so developing an understanding of the relationship between AI algorithms and techniques and their wider social uses—both positive and negative—is ever more critical. What we are pointing to here is the importance of a *constructionist paedagogy* in helping teachers and students to explore the 'real-world' consequences of their creations and to better understand the (societal) implications of AI.

However, while AI4K12 and many similar efforts are supportive of AI student programming projects, their focus is on reading, videos, discussions, and interactive demos. For example, in Touretzky et al. (2019), while they list several good activities, they do not suggest

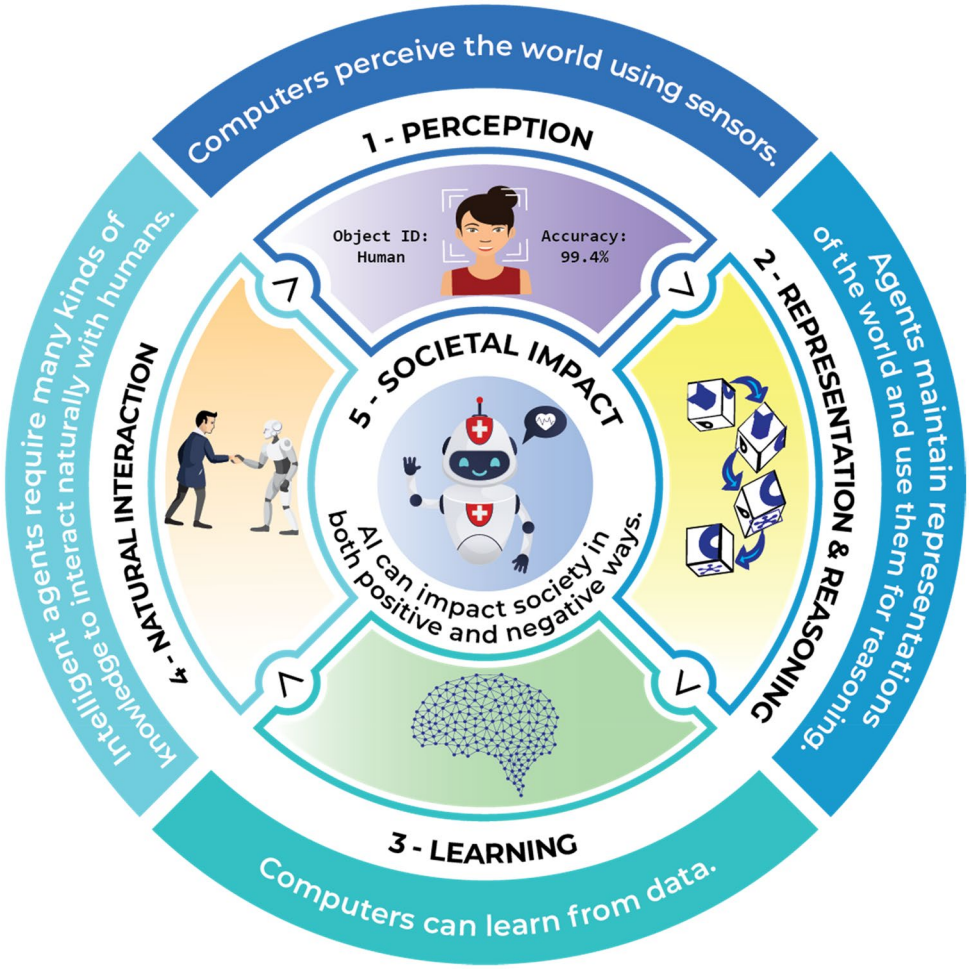

FIGURE 1 The five big ideas in AI according to AI4K12.org. Reproduced without modification from https://bit.ly/ai4k12-five-big-ideas [Colour figure can be viewed at wileyonlinelibrary.com]

constructionist-style projects until high school. And even then the suggestions are more like class assignments than student-led projects.

Here is a typical list of their suggestions:

> Students in K-2 should be able to describe the types of requests an intelligent assistant understands, and use a web app to demonstrate facial expression recognition. In grades 3-5 students should be able to distinguish a chatbot from a human, and analyse natural language examples to determine which ones would be difficult for a computer to understand, and why. In grades 6-8 students should be able to use parser demos to demonstrate syntactic parsing of sentences, and construct sentences that purely syntactic parsers will mishandle due to problems such as erroneous prepositional phrase attachment (e.g., 'I pour syrup for pancakes from a bottle'). They should also be able to show how parsers that take semantic information into account do a better job resolving attachment problems.
>
> In grades 9-12 students should be able to construct context-free grammars to parse simple languages, and use language processing tools to construct a chatbot. They should also be able to use sentiment analysis tools to extract emotional tone from text.

There are many opportunities for the constructionist projects connected to all the 'big ideas in AI' as we discuss below. With the appropriate tools and resources children in K-8 can also explore and use AI in their projects. No need to wait until grade 9 to introduce AI programming.

AI4K12, amongst many others, encourages teachers to have their students use tools such as Google's Teachable Machine (Google, 2017, 2019). Students can create a few classes of input, provide some examples, and experiment with the resulting deep learning model. Input can be images, sounds, or body poses. It is a wonderful web app but it is only a step towards constructionism. Students can *construct* recognition models of their own design and perform experiments with their models. They can even save and share their models. Students familiar with JavaScript or Python can then build a project around their trained classifier. Rarely, however, do we see any AI teaching guidance, suggesting that school students could build a project around a model they trained.

More constructionist alternatives to the Teachable Machine can be found in Machine Learning for Kids (Lane, 2020), eCraft2Lean's AI Snap! library (Kahn & Winters, 2018), mBlock Scratch-like programming system (mBlock, 2020), Cognimates (Druga, 2018), or the MIT App Inventor (Zhu, 2019). In addition to the similar functionality to the Teachable Machine (often inspired by Teachable Machine), these applications support non-expert programmers in embedding their trained classification models into their projects.

However, it is clear that teachers need to be at the centre of such endeavours and frameworks are needed (see Petrovai (2018) for the details on the development of the Teacher Adoption of Digital Technologies framework) to support their engagement with technology-based interventions.

AI programming systems designed for learning by children

Today there are several programming languages designed for learning that have been enhanced to support AI programming:

- The MIT App Inventor provides a very rich set of components and scripting blocks for constructing mobile device apps. In 2019 several extensions were developed that support image recognition and detection, sound recognition, sentiment analysis, optical character recognition, speech recognition, and more (Tang, 2019; Van Brummelen, 2019; Zhu, 2019). Several AI tutorial lessons have been developed (MIT App Inventor, 2020).
- Machine Learning for Kids provides an interface for training a neural network to classify images, sounds, text, or numbers. The trained model can then be used in a modified version of Scratch. The website contains dozens of project tutorials. It has been used by thousands of schools, code clubs, and families around the world (Wakelet, 2020).
- mBlock is a Chinese variant of Scratch that has an AI extension to connect to AI cloud services and a machine learning extension supporting the training and use of neural networks (mBlock, 2020).
- ML2Scratch is an open-source project by Junya Ishihara that provides an extension to a modified version of Scratch to access the functionality of a large number of pre-trained deep learning models (Ishihara, 2020).
- eCraft2Learn's Snap! AI blocks are described below.

These systems have in common the goal of making AI programming accessible to ordinary school students. Students can use these tools to build projects that see, listen, learn, classify, converse, and predict, thereby engaging, in a hands-on manner, with AI strengths, weaknesses, concepts and capabilities.

A closer look at one AI programming for children project

eCraft2learn was a European research project from 2016 to 2018. It focussed on bringing ideas and technology from the Maker Movement to STEAM education. As part of the project, the University of Oxford developed Snap! blocks for AI programming. They have continued to develop the blocks, guides, project suggestions, and sample projects after the eCraft2Learn project is finished. They recently added blocks for defining neural networks, training them, and using them for prediction and classification (Kahn et al., 2020). We are unaware of any other programming language designed for children and learning that supports high-performance deep machine learning.

eCraft2Learn partners in Greece and Finland field tested the blocks and learning resources. Workshops using the blocks have been run in Sri Lanka, Singapore, and Indonesia. Collaborators at the Beijing Normal University have begun efforts to test these resources with both school children and non-CS major university students. Two groups in India are developing lesson plans using these resources.

A unique aspect of this set of programming tools and resources is that it is completely web-based and largely server-free. There is no need to register or log in. By relying upon Google's TensorFlow.js (Smilkov et al., 2019) the machine learning occurs in the user's device (laptop, tablet, or phone). It runs at competitive speeds to other technologies since it can accelerate computations by using the device's graphical processing unit (GPU). Nothing needs to be installed. No data need to be transmitted to servers thereby enhancing privacy. It can run solely upon the local file system when Internet connections are not available or reliable. The Snap! blocks that use pre-trained neural networks and those for creating and training models all rely upon TensorFlow.js to perform all computations locally. Apps using the blocks can be very responsive since they avoid any latency due to communication overhead with servers. Unlike efforts to provide some of this kind of functionality in Scratch or the MIT App Inventor, the AI Snap! blocks work in the standard unmodified Snap! release.

The ethics of AI and constructionism: Two examples

The Snap! blocks we developed for AI programming within eCraft2Learn allow students to programme deep machine learning using neural network for prediction and classification (Kahn et al., 2020). These make the idea for exploring the ethical uses of training databases and to explore questions of how biases in such databases can lead to unethical outcomes that are racist and sexist. One example is the fundamental error a system will make trying to recognise a black face when trained on a database of white faces. Students would then discuss how such racist training databases, even when containing millions of training images, could be allowed to be developed and used, which includes a discussion on how to make AI more ethical. This leads to discussions about real-world incidents—including the withdrawal of the Tiny Image database (CSAIL, 2020; Prabhu & Birhane, 2020) and algorithmic injustices more broadly (Birhane & Cummins, 2019).

Embedding words in a high-dimensional space is a core technique in today's natural language processing efforts. This is generated by processing huge amounts of text (e.g. all of Wikipedia and/or billions of web pages). The eCraft2Learn Snap! library includes several new blocks to enable students to use word embeddings in their projects (Kahn et al., 2020). Amongst those projects are ones that search for biases in the data that was used to generate the word embeddings. For example, word embeddings can be used to solve puzzles like 'Man is to woman as doctor is to X'. Early implementations responded with 'nurse' due to societal biases reflected on the web. The Snap! word embedding blocks can also be used to construct programmes that determine if particular words associated with genders, ethnic groups, sexual orientations, etc., are unfairly closer to clusters of unpleasant concepts than to pleasant words (Caliskan et al., 2017).

A response to criticism of constructionism

Ames (2018) expresses a critique of constructionism, the most relevant of which is the following discussion on the potential equivalence between human cognition and computation:

> *Papert described cybernetics rapturously in Mindstorms as a potent framework for understanding learning by thinking about brains like we think about computers. This was not unusual: at the time, there was widespread belief that human brains were particularly sophisticated computers; Minsky, in fact, famously called them 'meat machines'. While the problems with this equivalence eventually contributed to the collapse of cybernetics and to the 'artificial intelligence winter' during which the field stagnated for decades, fragments of the belief persisted*. (p. 18:14)

While one can disagree with this view of AI history (and we do), the important question regarding the value of introducing AI programming to children is whether there really are 'problems with this equivalence'? Indeed, if one rejects the idea that thoughts are computations and brains are computational machines then one might question the value of helping children to construct AI programmes. If thoughts are something other than computations (in contrast to research in cognitive science and neuroscience (e.g. Miłkowski, 2017), then students may acquire a false mechanical concept of their thought processes. But what models of thinking might children acquire instead? Papert considered the alternative to be harmful 'Pop-Ed theories' (Papert, 1971). These are the theories that children do acquire about thinking. One is the 'blank-mind' theory that what one should do is make your mind a blank and wait for an idea to come. Another Papert named 'getting-it' where one expects understanding to come in a flash, the opposite of the Piagetian process of accommodation. Another common theory that interferes with

learning Papert called 'faculty'. For example, some people have the faculty of mathematics and some do not. Failure is because the problem is too hard to address given one's innate faculties. Our hope is that by engaging in AI programming students might acquire models of their thinking and learning that, while crude, can help them become better thinkers and learners.

Another criticism is that AI is based upon an obsolete and limited conception of cognition (Rescorla, 2020; Strube, 2001). The argument is that the old view of cognition underlying AI ignores the situated and social nature of cognition. The new view is that 'cognitive systems are [to be] conceived [of] as autonomous social agents, situated in a complex dynamic environment' (Strube, 2001). A constructionist response to this might be 'Yes, let's support learners with tools enabling them to build intelligent agents with robotic bodies (or simulated ones) that communicate and coordinate with other agents (and humans)'. Most of the AI programming tools we have described are capable of controlling robotic artefacts and communicating with other agents. Communication can be messages sent over a network or accomplished by relying upon the fact that the agents can see and listen to each other. Expanding AI programming by children to include the programming of 'autonomous social agents, situated in a complex dynamic environment' is an exciting future research direction.

POSSIBLE FUTURES

The early history of AI and constructionism was focussed on symbolic AI ('good-old-fashion AI'). More recently the focus has shifted greatly towards machine learning with neural networks. This is good since it provides students with very powerful capabilities enabling them to creatively build very capable artefacts. Students now have the tools to build apps that have the potential to make a positive difference in their community or the entire world (Tissenbaum et al., 2019). Students can acquire a first-hand experience with a technology that is rapidly changing the world. It is a technology they can use to become scientists, engineers, business executives, artists, musicians, doctors, or simply engaged citizens.

But something has been lost in focussing on neural networks. Aaron Sloman wrote of students making 'toy minds'. Minsky and Papert spoke of acquiring concepts to make one better at reflection and learning. But with neural nets, students are now making 'toy brains'. They are not programming at the level of concepts, symbols, and plans, but instead are relying upon crude approximations to how neurons work in brains. Moreover, as neural approaches are the ones that raise the clearest ethical issues, these provide critical learning opportunities for students to discuss the ethical issues by the use and misuse of AI (Benjamin, 2019).

An ideal future would involve a synthesis of these two schools of AI. Students could not only choose between computational building blocks that are symbolic or neural but use hybrids (a current topic of AI research). When the symbolic-neural computation is well-enough understood we should provide accessible hybrid building blocks to learners. Studies are needed to explore more closely whether, indeed, learners' hands-on exposure to AI and machine learning concepts has any effect on their self-reflection and learning skills. And whether the effects are evident regardless of whether the AI is symbolic or neural. Another open question is the minimum level of intellectual ability a child needs to benefit from AI programming.

AI is also being used in non-constructionist ways in schools. Automatic grading systems are proliferating. Learning analytics are applied to data generated from online courses. Research on intelligent tutoring systems continues to improve.

Currently these uses of AI in teaching work best when the software knows what the students were assigned to do. Maybe someday AI systems will be able to help guide students doing their own projects. Maybe AI systems will be able to provide suggestions and nudges that help students do projects without providing so much guidance that students are deprived of the joy and passion of self-directed creation of artefacts.

CONFLICT OF INTEREST

We declare there are no conflicts of interest.

ETHICS STATEMENT

This paper is a historical and theoretical description of interactions between AI and Constructionism. Hence there is no data, nor any need for ethical approval.

DATA AVAILABILITY STATEMENT

The software we developed that is described here is freely available at github.com/ecraft2learn/ai.

ORCID

Ken Kahn https://orcid.org/0000-0002-8208-7423
Niall Winters https://orcid.org/0000-0001-8597-2914

REFERENCES

Ames, M. G. (2018). Hackers, computers, and cooperation. *Proceedings of the ACM on Human-Computer Interaction*, *2*(CSCW), 1–19. https://doi.org/10.1145/3274287

Benjamin, R. (2019). *Race after technology: Abolitionist tools for the new Jim Code*. John Wiley & Sons.

Biesta, G. (2020). Have we been paying attention? Educational anaesthetics in a time of crises. *Educational Philosophy and Theory*, 1–3. https://doi.org/10.1080/00131857.2020.1792612

Birhane, A., & Cummins, F. (2019). Algorithmic injustices: Towards a relational ethics. *arXiv Preprint arXiv:1912.07376*.

Caliskan, A., Bryson, J., & Narayanan, A. (2017). Semantics derived automatically from language corpora contain human-like biases. *Science*, *356*(6334), 183–186. https://doi.org/10.1126/science.aal4230

CSAIL. (2020). https://groups.csail.mit.edu/vision/TinyImages/

Druga, S. (2018). *Growing up with AI: Cognimates: from coding to teaching machines* (Doctoral dissertation). Massachusetts Institute of Technology.

Goldstein, I., Lieberman, H., Bochner, H., & Miller, M. (1975). LLOGO: An implementation of LOGO in LISP. MIT Artificial Intelligence Lab Memo Number 307a.

Google. (2017). Teachable machine. https://experiments.withgoogle.com/ai/teachable-machine

Google. (2019). https://blog.google/technology/ai/teachable-machine/

Google. (2020). Do-it-yourself artificial intelligence. https://aiyprojects.withgoogle.com/

Ishihara, J. (2020). https://github.com/champierre/ml2scratch

Kahn, K. (1975). A Logo natural language system. Technical Report, MIT AI Lab, December. LOGO Working Paper 46.

Kahn, K. (1976). A knowledge-based computer animation system. Technical report, MIT AI Lab, February, LOGO Working Paper 47, AI Working Paper 119.

Kahn, K. (1977). Three interactions between AI and education. In E. Elcock, & D. Michie (Eds.), *Machine intelligence 8: Machine representations of knowledge*. Ellis Horwood Ltd. and John Wylie & Sons.

Kahn, K. (1984). A grammar kit in prolog. In M. Yazdani (Ed.), *New horizons in educational computing*. Halsted Press.

Kahn, K., Lu, Y., Zhang, J., Winters, N., & Gao, M. (2020). Deep Learning Programming by All. *Proceedings of Constructionism Conference 2020*, Dublin, Ireland.

Kahn, K., Lu, Y., Zhang, J., Winters, N., & Gao, M. (2020). Programming word embeddings in Snap!

Kahn, K., & Winters, N. (2017, September). Child-friendly Programming Interfaces to AI Cloud Services. *Proceedings of the EC-TEL 2017 ConferenceTallinn*, Estonia.

Kahn, K., & Winters, N. (2018, August). AI Programming by Children. *Constructionism Conference*, Vilnius, Lithuania.

Kowalski, R. A. (1984). Logic as a computer language for children. In M. Yazdani (Ed.), *New horizons in educational computing*. Halsted Press.

Kowalski, R. A. (2015). A short story of my life and work. http://www.doc.ic.ac.uk/~rak/history.pdf

Lane, D. (2020). Machine learning for kids. https://machinelearningforkids.co.uk/#!/about

Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016) *Intelligence unleashed. An argument for AI in education*. Pearson.

Marques, L. S., Von Wangeheim, C. G., & Hauck, J. C. (2020). Teaching machine learning in school: A systematic mapping of the state of the art. *Informatics in Education*, *19*(2), 283–321.

mBlock. (2020). https://www.mblock.cc/doc/en/part-one-basics/overview.html

Miłkowski, M. (2017). Objections to computationalism. A short survey. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society* (pp. 2723–2728). Computational Foundations of Cognition.

Miller, M. L. (1979). A structured planning and debugging environment for elementary programming. *International Journal of Man-Machine Studies*, *11*(1), 79–95.
Minsky, M. (1988). *Society of mind*. Simon and Schuster.
Minsky, M. (2009). OLPC Memo 5: Education and Psychology. https://web.media.mit.edu/~minsky/OLPC-5.html
Minsky, M. (2019). In C. Solomon, & X. Xiao (Eds.) *Inventive minds: Marvin Minsky on education*. MIT Press.
MIT App Inventor. (2020). http://appinventor.mit.edu/explore/ai-with-mit-app-inventor
Nichol, J., & Dean, J. P. (1984). Computers, and history. In M. Yazdani (Ed.), *New horizons in educational computing*. Halsted Press.
Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings. Learning Cultures and Computers*. Springer.
Papert, S. (1971). Teaching children thinking. MIT Artificial Intelligence Lab Memo Number 247.
Papert, S., & Harel, I. (1991). *Constructionism*. Ablex Publishing Corporation.
Papert, S., & Solomon, C. (1971). Twenty things to do with a computer. *MIT Artificial Intelligence Lab Memo Number*, *248*.
Petrovai, K. (2018). *A study investigating teachers' use of, and views on, tablets in the teaching of mathematics* (PhD thesis). University of Oxford.
Prabhu, V. U., & Birhane, A. (2020). Large image datasets: A pyrrhic win for computer vision?. *arXiv preprint arXiv:2006.16923*.
Rescorla, M. The computational theory of mind. In E. N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy* (Spring 2020 Edition). https://plato.stanford.edu/entries/computational-mind/#ArgAgaCom
Sergot, M. (1984). A query-the-user facility for logic programming. In M. Yazdani (Ed.), *New Horizons in Educational Computing*. Halsted Press.
Sloman, A. (1984). Experiencing computation: A tribute to Max Clowes. In M. Yazdani (Ed.), *New horizons in educational computing*. Halsted Press.
Sloman, A. (2012). Examples of general and AI programming teaching materials illustrated using Poplog/Pop-11. http://www.cs.bham.ac.uk/research/projects/poplog/examples/
Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., & Zhang, K. (2019). *TensorFlow.js: Machine learning for the web and beyond*.
Solomon, C., Harvey, B., Kahn, K., Lieberman, H., Miller, M., Minsky, M., Papert, A., & Silverman, B. (2020). History of logo. *Proceedings of the ACM on Programming Languages*, *4*(HOPL), 1–66. https://doi.org/10.1145/3386329
Strube, G. (2001). Cognitive science: Overview. *International Encyclopedia of the Social & Behavioral Sciences*, 2158–2166. https://www.sciencedirect.com/science/article/pii/B0080430767014418#s0090
Tang, D. (2019). Empowering novices to understand and use machine learning with personalized image classification models. *intuitive analysis tools, and MIT App Inventor* (Master's thesis). Massachusetts Institute of Technology.
Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM*, *62*(3), 34–36.
Touretzky, D. (2020). Ai4k12 wiki. https://github.com/touretzkyds/ai4k12/wiki
Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (2019). Envisioning AI for K-12: What should every child know about AI?. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. *33*, pp. 9795–9799). https://doi.org/10.1609/aimag.v40i4.5289
Van Brummelen, J. J. R. (2019). *Tools to create and democratize conversational artificial intelligence* (Master's thesis). Massachusetts Institute of Technology.
Wakelet. (2020). https://wakelet.com/wake/4d3680cb-2a11-49ab-9180-fcfeced48393
Winters, N., Eynon, R., Geniets, A., Robson, J., & Kahn, K. (2020). Can we avoid digital structural violence in future learning systems? *Learning Media and Technology*, *45*(1), 17–30.
Wolfram, S. (2017). Machine learning for middle schoolers. https://writings.stephenwolfram.com/2017/05/machine-learning-for-middle-schoolers/
Yazdani, M. (1984). Editor, *New Horizons in Educational Computing*. Halsted Press.
Zhang, Y., Wang, J., Bolduc, F., & Murray, W. (2019). LP Based Integration of Computing and Science Education in Middle Schools, *Proceedings of the ACM Conference on Global Computing Education* (CompEd '19). Association for Computing Machinery, New York, NY, USA, 44–50. https://doi.org/10.1145/3300115.3309512
Zhu, K. (2019). *An educational approach to machine learning with mobile applications* (Master's thesis). Massachusetts Institute of Technology.

How to cite this article: Kahn, K., & Winters, N. (2021). Constructionism and AI: A history and possible futures. *British Journal of Educational Technology*, 52, 1130–1142. https://doi.org/10.1111/bjet.13088