Learning Tools Using Block-based Programming for AI Education

Chris-Bennet Fleger
Faculty of Electrical Engineering and Computer Science
Leibniz University Hannover
Hannover, Germany
https://orcid.org/0000-0001-5910-6772

Yousuf Amanuel
Faculty of Electrical Engineering and Computer Science
Leibniz University Hannover
Hannover, Germany
https://orcid.org/0000-0002-7286-4495

Johannes Krugel
Faculty of Electrical Engineering and Computer Science
Leibniz University Hannover
Hannover, Germany
https://orcid.org/0000-0003-0967-4872

2023 IEEE Global Engineering Education Conference (EDUCON) | 979-8-3503-9943-1/23/$31.00 ©2023 IEEE | DOI: 10.1109/EDUCON54358.2023.10125154

***Abstract*—This work identifies the capabilities of a block-based programming approach for learning machine learning concepts. It focuses on the following overarching research question: "How can block-based programming tools be used to facilitate the understanding and application of machine learning concepts in K-12 education?".**

To answer this question, guidelines for conducting a systematic literature review are followed, resulting in the study of 17 different learning tools. These tools are examined for their technical nature, content coverage, design features, intelligibility, evaluations, and deployability.

The findings suggest that the vast majority of tools focus on a high-level representation of classification models that children can create in an extended version of the Scratch programming environment. By this, however, only one facet of machine learning is addressed, and deeper insights into the underlying functions are not provided. In addition, technical, linguistic, and conceptual barriers to the design of tools and the wider curricula become apparent.

***Index Terms*—Computer science education, Computer Uses in Education, Learning environments, Artificial Intelligence, Machine learning**

I. Introduction

Block-based programming (BBP) could become helpful in learning artificial intelligence (AI) due to its ability to visualize programming code, making it more tangible for learners. BBP has helped to increase students' computational thinking (CT) skills [1, 2]. Coding environments like Scratch or Snap! offer ways of implementing constructionist projects with high complexity [3]. In contrast to text-based programming environments, they can display the program structure using a building block metaphor, making it more tangible for children [4]. Due to the physical constraint of the blocks, syntactic errors can be prevented. In this way, the focus is shifted toward the semantics of the program. In addition, users can use the colorful command palettes, so they do not need prior knowledge of how programming works [4]. These benefits might show merit when proposing learning models for AI-related topics, which is why a selection of relevant papers will be further examined.

Long and Magerko introduce the definition of AI literacy, which describes the development of competencies which enable an individual to critically evaluate and understand AI, instead of merely making use of AI systems without further knowledge of how they operate [5]. One of these competencies is specifically targeted at machine learning (ML) since it is an essential subfield of AI with applications ranging from social media to healthcare [5]. A critical ethical issue presented in the paper is "Transparency". The authors explain that ML algorithms are usually black boxes where the functionalities remain opaque to users, potentially causing deception and misunderstanding [5]. Thus, we seek to investigate block-based programming tools in terms of their suitability to open these black boxes of ML. The general research question of our review is "How can block-based programming tools be used to facilitate the understanding and application of machine learning concepts in K-12 education?" This general question is decomposed into six research questions as follows:

- **RQ1:** *What BBP tools for learning ML exist?*
- **RQ2:** *Which ML concepts are covered by the BBP tools?*
- **RQ3:** *How do the design characteristics of different BBP environments affect the learning of ML?*
- **RQ4:** *How age-appropriate are the BBP tools for students?*
- **RQ5:** *How was the use of the BBP tools for learning ML evaluated?*
- **RQ6:** *How do the BBP tools provide insights into the black box of ML?*

These questions discuss the proposed tools' technical nature, content coverage, design features, intelligibility, evaluations, and deployability. With this research objective, we aim to investigate the suitability of existing block-based programming tools for ML education to support the design of future tools.

II. Related Work

This work deals with AI education, specifically ML education on the one hand and block-based programming on the other, and looks at how the two can be reconciled. Our work builds on a review by Wangenheim et al. from 2020 [6]. In a ten-year (2010-2020) systematic mapping, 16 visual tools were analyzed and their characteristics were studied. Among the tools presented were six block-based and ten

flow-based tools. The examined characteristics were those concerning the ML platform and deployment platform, as well as educational characteristics. In addition, they looked at the development and evaluations of the tools. The results of this mapping indicate that visual tools can be beneficial to ML instruction in K-12 and can make a valuable contribution, especially for novices. In contrast to their study, our work deals specifically with block-based programming environments and includes additional tools, some of which have only emerged in the last two years. The research questions of this thesis are more directed at the content level by investigating which ML aspects are being covered. Furthermore, it will also be examined how age-appropriate the tools are in terms of ease of entry and visual appeal. In addition, the various BBP environments used will be compared with each other based on their design elements. A particular focus is whether the black box of machine learning can genuinely be broken up for students. While the basics of ML, such as classification, can be illustrated using technological equipment such as motion sensors or cameras, for a deeper understanding it is vital to gain insight into how the algorithms work.

III. Methodology: Systematic Literature Review

Our work provides a systematic analysis on the current state of research on AI education in the context of block-based programming that offers a high level of information so that further research can build on these results. To achieve this goal, guidelines by the "Preferred Reporting Items for Systematic Reviews and Meta-Analyses" 2020 statement (PRISMA) was consulted for writing a systematic literature review (SLR) [7]. In this work, we closely followed those guidelines to ensure that the work process is intelligible and that the literature analysis is structured and reproducible. The literature for this review has been collected from the following four sources: ACM Digital Library, IEEE Xplore Digital Library, ScienceDirect, and Google Scholar. The search was conducted from May 10 to May 17, 2022. These databases were selected due to their high relevance in the field of computing [6, 8]. Access to the publications was provided by our university library. In addition to the primary search in the digital databases, a backward snowballing phase, as proposed by Wohlin, was initiated after the results had been gathered to find further potentially relevant articles [9].

For the screening process, we first established eligibility criteria and formulated search strings for the databases. We excluded papers for the following reasons:

- *The language of the paper is neither English nor German*
- *The target group exceeds the level of K-12 students*
- *The paper was published before 2010*
- *The paper contains only one page or merely an abstract*
- *The paper is not included in the university subscription*

We then created four different keyword sets that include (sub)terms and synonyms from the categories of learning, school, artificial intelligence, and block-based programming. Using Boolean operators, we constructed a comprehensive search string that we ran in the databases. For Google Scholar, we had to truncate the string considerably to obtain a suitable number of search results, which left us with the following:

(education OR educational OR school OR children OR kids OR curricula) AND "machine learning" AND "block-based programming".

After entering these search strings, we then reviewed the search results successively. In the first screening process, the title, publication date, type of source, and the number of pages were checked by hand. Subsequently, the results were skimmed, and it was determined if any eligibility criteria were violated. The abstract was considered first, before the paper was read. If the paper seemed relevant after the first screening process, it was imported into reference management software. *Citavi*[1] was used for this purpose. The inclusion was noted in a text file for the corresponding database together with the preceding search string. A maximum of 100 search results were examined, as the relevance of the papers quickly decreased. The papers included in *Citavi* were examined in more detail in a second screening process. The papers were read thoroughly, and it was investigated whether the research questions could be answered. In this context, it was necessary to check whether the tool actually used a block-based programming environment. The paper was finally considered relevant if the focus lay on learning ML content and the tool's target audience included K-12 students.

After the search was performed in all four databases, the backward snowballing phase was initiated. The reference lists of included papers were processed sequentially by looking at the titles. If the title of a reference appeared to fit the research topic, it was then checked where in the paper it was cited to make assumptions about the content. If the citation seemed relevant, the source was subsequently inspected. If a link was provided, it was used. Otherwise, Google Scholar was used for the search. If these papers were eventually found to be relevant, they were included. In this way we were able to include three more papers, resulting in a total set of 17 tools. We documented the results of our review in a table where the answers to the research questions are presented in separate columns.

IV. Results

Applying the methods for conducting an SLR, as proposed by PRISMA, resulted in the analysis of 17 different learning tools. The results are shown in Table I. The research questions can be answered as follows:

RQ1: The block-based coding environment is often just a part of the whole application. Additional websites and interfaces correspond to more steps students must complete before they begin coding. These steps include getting access to the coding environment through registration and connection

[1]https://www.citavi.com/

processes and setting up ML models by training them using various input devices. Looking at the programming environment per se, the majority of the tools use an extended version of Scratch. These extensions provide a concise amount of ML code blocks that let the user use, modify, and create machine learning models. However, the tools by Estevez et al. and Michaeli et al. (*SnAIp*) are the only ones that encourage the user to build the ML models themselves, using none other than the standard programming blocks available in Snap! and Scratch. Concerning the technical nature of the tools, conventional programming frameworks and powerful APIs like *Tensorflow.js* have been used instead of building everything from the ground up, and some tools run entirely client-side.

RQ2: Regarding the tools' coverage of ML topics, there is a strong imbalance in the choice of topics. There are many more tools that represent supervised learning (SL) than those that represent unsupervised learning (UL) or reinforcement learning (RL). Along with that, the majority of tools corroborate the idea of ML with the concept of classification. Models and algorithms are hardly discussed. It is also striking that concepts like decision trees do not appear in any of the tools. The concepts that are included appear to be relevant as they are featured in ACM and IEEE's Computer Science Curricula of 2013 [10] or the book "Artificial intelligence: A modern approach" by Russel and Norvig [11], for instance. Tools that display a lot of details about the ML models present abstractions that are suitable for children. However, some of these abstractions discard crucial aspects of the ML model which can lead to deception about the actual functionality of the model.

RQ3: All analyzed tools appear to be well-designed. Despite a few tools having chosen an unfavorable palette of code block colors that cannot be changed, there are hardly any characteristics that stand out as irritating or confusing. Some tools have put a little more effort into their design than others which can be observed from aspects like uploaded icons, individual sprites, and translations. Regarding the BBP environments, Scratch seems to have the best usability due to fluent animations, easy language, and strong support in navigation with the environment. Nevertheless, features like the alerts in App Inventor are also valuable and contribute to the other programming environments being considered well-designed. One advantage that *BlockWiSARD*, *PRIMARY AI*, and *Milo* have over the other environments is that they were developed specifically for AI and data science education. As a consequence, there are no additional blocks or interface components that could be distracting from the ML task. In contrast, App Inventor has the most complicated GUI, which becomes relevant when considering the age-appropriateness of the tools.

RQ4: Tools with higher educational requirements are targeted at older students accordingly. The used terminology varies among the tools. Confronting children with complicated terms like "API keys" should be avoided as it could hinder their learning progress. Apart from that, Scratch tools have a seemingly *lower floor* [32] than tools using App Inventor

Name of the Tool	Author(s)	Programming Environments	ML Subfield(s)	ML Concept(s)	Target Groups	Test Groups
ML4K	D. Lane, 2018 [12]	Scratch, App Inventor	SL	CL	all students	–
Cognimates	S. Druga, 2018 [13]	Scratch	SL	CL	primary & middle school s.	primary & middle school s.
Tooee	Y. Park and Y. Shin, 2021 [14]	Scratch	SL	CL	all students	primary school teachers
Tool by Reddy et al.	T. Reddy et al., 2021 [15]	Scratch	SL	CL	middle school students	middle school students
DeepScratch	N. Alturayeif et al., 2020 [16]	Scratch	SL	CL, ANN	all students	all students
AIpacaML	A. Zimmermann-Niefield et al., 2020 [17]	Scratch	SL	CL	middle school s.	primary & middle school s.
Scratch Nodes ML	A. Agassi et al., 2019 [18]	Scratch	SL	CL	primary school students	–
Tool by Estevez et al.	J. Estevez et al., 2019 [19]	Scratch	SL, UL	ANN, KMC	high school students	high school students
SnAIp	T. Michaeli et al., 2019/2020 [20, 21]	Snap!	UL, RL	CL, QL	middle & high school students	–
Milo	A. Rao et al., 2018 [22]	Milo	SL, UL	CL, ANN, KMC, KNN	high school & undergraduate s.	undergraduate students
PoseBlocks	B. Jordan et al., 2021 [23]	Scratch	SL	CL	all students	middle school students
eCraft2Learn	K. Kahn et al., 2018/2020 [24–26]	Snap!	SL	CL	high school students	high school students
Tool by Olari et al.	V. Olari et al., 2021 [27]	NEPO	SL, RL	ANN, QL	all students	all students
PRIMARY AI	S. Lee et al., 2021 [28]	PRIMARY AI	SL	CL	primary & middle school s.	–
LearningML	J. D. Rodríguez-García et al., 2020 [29]	Scratch	SL	CL	all people	undergraduate students
BlockWiSARD	R. Queiroz et al., 2020 [30]	BlockWiSARD	SL	CL	all people	–
Tool by Zhu	K. Zhu, 2019 [31]	App Inventor	SL	CL	middle & high school students	high school students

TABLE I
TOOLS FOR ML EDUCATION COMPARED BY PROGRAMMING LANGUAGE, CONTENT ASPECTS, AND TARGET/TEST GROUPS; CL:=CLASSIFICATION, SL:=SUPERVISED LEARNING, UL:=UNSUPERVISED LEARNING, RL:=REINFORCEMENT LEARNING, ANN:=ARTIFICIAL NEURAL NETWORKS, KNN:=K-NEAREST NEIGHBORS, KMC:=K-MEANS CLUSTERING, QL:=Q-LEARNING

and Snap!. This gap is due to Snap! being a more powerful version of Scratch that allows for advanced programming. Simultaneously, App Inventor uses a more complex environment introducing the concept of Components that users need to grasp before writing programs. Tools that let children build their classifiers have *wider walls* [32] than tools that require them to complete an ML algorithm.

RQ5: The evaluations show that the BBP tools can foster students' understanding of machine learning. The authors seem to have developed engaging and easy-to-grasp tools. Challenges include translating the tools into different languages, preparing the students adequately for the learning environment, and overcoming technical hurdles for successful deployment. However, for more significant outcomes, it would be necessary to conduct more extensive studies with more participants. The participants should ideally reflect the whole age spectrum of K-12 students to assess which tools are too complicated for younger students. It also remains unclear to what extent the tools could promote the learning effect per se, given that participants already had prior programming experience and were supported and guided by teachers during the sessions.

RQ6: Most tools seem to neglect the relevance of breaking open the black box of ML. Their algorithms are obscure, and the user has no chance to fully grasp, for instance, how predictions by the ML models they created are calculated. Some tools try to counteract this by allowing the user to modify the values of the model. Nevertheless, only two out of 17 tools reveal the entire implementation of the underlying algorithms, which is a considerably small portion. According to Long and Magerko, this opaqueness of underlying functionalities can lead to deception and misunderstanding, which presents a severe issue [5].

V. Discussion and Conclusion

Our findings suggest that many similar block-based programming tools exist for learning ML. The data obtained from the literature review shows an uneven distribution of tool types. In line with Wangenheim et al., the majority of the findings propose classifier-building tools using an extended version of Scratch [6]. These tools appear to be engaging and easy to use regarding the evaluations. Furthermore, they could foster students' understanding of machine learning. However, they only provide a high-level view of the content and do not break open the black box of ML. Only a few tools give more insight into ML concepts, also covering unsupervised and reinforcement learning content. The focus lies on illustrating supervised learning aspects, with classification being the dominant concept. Other concepts, such as decision trees, are not covered.

Of all the BBP environments used, Scratch seems to be very suitable for children, whereas Snap! and App Inventor are more appropriate for high school or undergraduate students. Less popular BBP environments also have potential, but more evaluations are necessary for stronger conclusions. The results indicate technical, linguistic, and conceptual barriers exist to deploying the tools in schools. Assuming these barriers are overcome and more comprehensive ML curricula support is provided by the tools, this analysis supports the theory that there is a great potential for using block-based programming in K-12 education to facilitate the learning of ML concepts. Based on these results, we are currently developing a learning tool that focuses on minimizing the disadvantages found in other tools. This can be achieved by simplifying the UI, reducing the number of blocks to only the essentials, and explaining the different algorithms step-wise.

References

[1] D. Bau, J. Gray, C. Kelleher, J. Sheldon, and F. Turbak, "Learnable programming: Blocks and beyond," *Commun. ACM*, vol. 60, no. 6, pp. 72–80, May 2017. DOI: 10.1145/3015455.

[2] N. Arslan Namli and B. Aybek, "An investigation of the effect of block-based programming and unplugged coding activities on fifth graders' computational thinking skills, self-efficacy and academic performance," *Contemporary Educational Technology*, vol. 14, no. 1, 2022. DOI: 10.30935/cedtech/11477.

[3] S. Jatzlau, R. Romeike, E. Jasut, and V. Dagien, "How high is the ceiling? applying core concepts of block-based languages to extend programming environments," *Constructionism 2018, Vilnius*, pp. 286–294, 2018.

[4] D. Weintrop, "Block-based programming in computer science education," *Communications of the ACM*, vol. 62, no. 8, pp. 22–25, 2019. DOI: 10.1145/3341221.

[5] D. Long and B. Magerko, "What is AI literacy? competencies and design considerations," in New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–16. DOI: 10.1145/3313831.3376727.

[6] C. Gresse von Wangenheim, J. C. R. Hauck, F. S. Pacheco, and M. F. Bertonceli Bueno, "Visual tools for teaching machine learning in k-12: A ten-year systematic mapping," *Education and information technologies*, vol. 26, no. 5, pp. 5733–5778, 2021. DOI: 10.1007/s10639-021-10570-8.

[7] M. J. Page *et al.*, "The prisma 2020 statement: An updated guideline for reporting systematic reviews," *Systematic Reviews*, vol. 10, no. 1, 2021. DOI: 10.1186/s13643-021-01626-4.

[8] I. T. Sanusi, S. S. Oyelere, F. J. Agbo, and J. Suhonen, "Survey of resources for introducing machine learning in k-12 context," in *2021 IEEE Frontiers in Education Conference (FIE)*, 2021, pp. 1–9. DOI: 10.1109/FIE49875.2021.9637393.

[9] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering - EASE '14*, M. Shepperd, T. Hall, and I. Myrtveit, Eds., New York, New York, USA: ACM Press, 2014, pp. 1–10. DOI: 10.1145/2601248.2601268.

[10] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society, *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: Association for Computing Machinery, 2013. DOI: 10.1145/2534860.

[11] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, Fourth edition, global edition, ser. Pearson series in artificial intelligence. Harlow: Pearson, 2022.

[12] L. Dale, "Machine learning for kids," 2018. [Online]. Available: https://machinelearningforkids.co.uk/#!/about (visited on 07/09/2022).

[13] S. Druga, "Growing up with AI: Cognimates: From coding to teaching machines," M.S. thesis, Massachusetts Institute of Technology, 2018. [Online]. Available: https://www.media.mit.edu/publications/growing-up-with-ai/ (visited on 07/26/2022).

[14] Y. Park and Y. Shin, "Tooee: A novel scratch extension for k-12 big data and artificial intelligence education using text-based visual blocks," *IEEE Access*, vol. 9, pp. 149 630–149 646, 2021. DOI: 10.1109/ACCESS.2021.3125060.

[15] T. Reddy, R. Williams, and C. Breazeal, "Text classification for AI education," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, M. Sherriff, Ed., ser. ACM Digital Library, New York, NY, United States: Association for Computing Machinery, 2021, p. 1381. DOI: 10.1145/3408877.3439689.

[16] N. Alturayeif, N. Alturaief, and Z. Alhathloul, "Deepscratch: Scratch programming language extension for deep learning education," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, 2020. DOI: 10.14569/IJACSA.2020.0110777.

[17] A. Zimmermann-Niefield, S. Polson, C. Moreno, and R. B. Shapiro, "Youth making machine learning models for gesture-controlled interactive media," in *Proceedings of the Interaction Design and Children Conference*, ser. ACM Digital Library, New York, NY, United States: Association for Computing Machinery, 2020, pp. 63–74. DOI: 10.1145/3392063.3394438.

[18] A. Agassi, H. Erel, I. Y. Wald, and O. Zuckerman, "Scratch nodes ml," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, S. Brewster, Ed., ser. ACM Digital Library, New York,NY,United States: Association for Computing Machinery, 2019, pp. 1–6. DOI: 10.1145/3290607.3312894.

[19] J. Estevez, G. Garate, and M. Grana, "Gentle introduction to artificial intelligence for high-school students using scratch," *IEEE Access*, vol. 7, pp. 179 027–179 036, 2019. DOI: 10.1109/ACCESS.2019.2956136.

[20] S. Jatzlau, T. Michaeli, S. Seegerer, and R. Romeike, "It's not magic after all – machine learning in snap! using reinforcement learning," in *2019 IEEE Blocks and Beyond Workshop (B&B)*, 2019, pp. 37–41. DOI: 10.1109/BB48857.2019.8941208.

[21] T. Michaeli, S. Seegerer, S. Jatzlau, and R. Romeike, "Looking beyond supervised classification and image recognition–unsupervised learning with snap!" *CONSTRUCTIONISM 2020*, pp. 395–402, 2020.

[22] A. Rao, A. Bihani, and M. Nair, "Milo: A visual programming environment for data science education," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2018, pp. 211–215. DOI: 10.1109/VLHCC.2018.8506504.

[23] B. Jordan, N. Devasia, J. Hong, R. Williams, and C. Breazeal, "Poseblocks: A toolkit for creating (and dancing) with AI," in *Proc. of the 11th Symp. on Education Advances in Artificial Intelligence (EAAI'21)*, vol. 35, 2021, pp. 15 551–15 559. DOI: 10.1609/aaai.v35i17.17831.

[24] K. Kahn and N. Winters, "AI programming by children," *Constructionism 2018*, pp. 322–331, 2018.

[25] K. Kahn, R. Megasari, E. Piantari, and E. Junaeti, "AI programming by children using snap! block programming in a developing country," in *EC-TEL Practitioner Proceedings 2018: 13th European Conference On Technology Enhanced Learning*, Springer, 2018. DOI: 10.1007/978-3-319-98572-5.

[26] K. Kahn, Y. Lu, J. Zhang, N. Winters, and M. Gao, "Deep learning programming by all," *Constructionism 2020*, pp. 281–292, 2020.

[27] V. Olari, K. Cvejoski, and Ø. Eide, "Introduction to machine learning with robots and playful learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 17, pp. 15 630–15 639, May 2021. DOI: 10.1609/aaai.v35i17.17841.

[28] S. Lee *et al.*, "AI-infused collaborative inquiry in upper elementary school: A game-based learning approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 15 591–15 599. DOI: 10.1609/aaai.v35i17.17836.

[29] J. D. Rodríguez García, J. Moreno-León, M. Román-González, and G. Robles, "Learningml: A tool to foster computational thinking skills through practical artificial intelligence projects," *Revista de Educación a Distancia (RED)*, vol. 20, no. 63, 2020. DOI: 10.6018/red.410121.

[30] R. Lacerda Queiroz, F. Ferrentini Sampaio, C. Lima, and P. Machado Vieira Lima, "AI from concrete to abstract," *AI & SOCIETY*, vol. 36, no. 3, pp. 877–893, 2021. DOI: 10.1007/s00146-021-01151-x.

[31] K. Zhu, "An educational approach to machine learning with mobile applications," M.S. thesis, Massachusetts Institute of Technology, 2019.

[32] M. Resnick and B. Silverman, "Some reflections on designing construction kits for kids," in *Proceedings of the 2005 conference on Interaction design and children*, 2005, pp. 117–122. DOI: 10.1145/1109540.1109556.