Education and Information Technologies (2021) 26:5733–5778
https://doi.org/10.1007/s10639-021-10570-8

Visual tools for teaching machine learning in K-12: A ten-year systematic mapping

Christiane Gresse von Wangenheim[1] · **Jean C. R. Hauck**[1] · **Fernando S. Pacheco**[2] · **Matheus F. Bertonceli Bueno**[1]

Received: 6 November 2020 / Accepted: 27 April 2021 / Published online: 1 May 2021

Abstract

Teaching Machine Learning in school helps students to be better prepared for a society rapidly changing due to the impact of Artificial Intelligence. This requires age-appropriate tools that allow students to develop a comprehensive understanding of Machine Learning in order to become creators of smart solutions. Following the trend of visual languages for introducing algorithms and programming in K-12, we present a ten-year systematic mapping of emerging visual tools that support the teaching of Machine Learning at this educational stage and analyze the tools concerning their educational characteristics, support for the development of ML models as well as their deployment and how the tools have been developed and evaluated. As a result, we encountered 16 tools targeting students mostly as part of short duration extracurricular activities. Tools mainly support the interactive development of ML models for image recognition tasks using supervised learning covering basic steps of the ML process. Being integrated into popular block-based programming languages (primarily Scratch and App Inventor), they also support the deployment of the created ML models as part of games or mobile applications. Findings indicate that the tools can effectively leverage students' understanding of Machine Learning, however, further studies regarding the design of the tools concerning educational aspects are required to better guide their effective adoption in schools and their enhancement to support the learning process more comprehensively.

Keywords Visual tool · Machine learning · Computing education · K-12

✉ Christiane Gresse von Wangenheim
c.wangenheim@ufsc.br

[1] Department of Informatics and Statistics, Federal University of Santa Catarina, Florianópolis, Brazil

[2] Department of Electronics, Federal Institute of Santa Catarina, Florianópolis, Brazil

1 Introduction

Machine Learning (ML) is implemented into many devices and services that are part of our everyday life, for example, recommendation services, healthcare diagnosis, or autonomous vehicles. Thus, to prepare citizens, including young people, to become responsible and conscientious users and creators of intelligent solutions, it is important to popularize a basic understanding of ML technologies (Kandlhofer et al., 2016; Touretzky et al., 2019a; Wong et al., 2020). Due to the growth of Artificial Intelligence (including Machine Learning), for many countries it has also become a major strategy to promote competitiveness, requiring more people to seek a career in AI (Forbes, 2019; Hiner, 2017).

Yet, teaching fundamental AI (including ML) concepts and techniques has traditionally been done only in higher education (Torrey, 2012). And, although computing education is increasingly being included in K-12 worldwide, these programs rarely cover AI content at this educational stage (Hubwieser, 2015), although studies have shown that children are able to learn ML concepts from a relatively young age (Hitron et al., 2019). The exposure to this kind of complex knowledge has even the potential to enhance children's everyday skills, better equipping them to deal with the social, economic, and ethical issues that are arising from the use of ML (Kahn et al., 2020). Furthermore, AI literacy may encourage more students to consider careers in this area and provide solid preparation for higher education and their future profession.,

According to AI4K12 (Touretzky et al., 2019a), AI education should cover five big ideas in K-12, including Machine Learning that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Touretzky et al., 2019a; Wollowski et al., 2016). This includes an understanding of basic ML concepts, such as learning algorithms and fundamentals of neural networks as well as the limitations and ethical concerns related to ML. And, for students to become not just consumers of AI, but creators of intelligent solutions, this also requires teaching the application of these concepts, e.g., by developing image recognition models, since students who experience in a hands-on manner the possibilities, strengths, and weaknesses of ML are more likely to obtain a deeper understanding (Kahn et al., 2018; Kandlhofer et al., 2016; Touretzky et al., 2019b). Therefore, primarily active learning that emphasizes doing and direct experience by the student is important, as it helps to make ML transparent, enabling students to build correct mental models, and encouraging them to develop their own ML applications aiming at an engaging education (Wong et al., 2020).

Yet, the development of ML-enabled applications in real-world settings is non-trivial and the development process differs from that of traditional software (Lwakatare et al., 2019). Developing an ML model involves several tasks from acquiring a (labeled) set of examples, selecting an appropriate learning algorithm and its parameters, training the model, and evaluating the model's performance (Lwakatare et al., 2019; Ramos et al., 2020). It requires an understanding of complex algorithms and working processes, as well as a constantly increasing zoo of

architectures, frameworks, etc., which makes choosing a suitable one a difficult task for novices (Gillies, 2016; Gutosk, 2017; Sulmont et al., 2019) as well as requiring the user to have a certain level of programming skills (Xie et al., 2019). As a consequence, students typically face several difficulties when starting to learn ML, making the process of building ML models inaccessible to many people (Ramos et al., 2020; Sankaran et al., 2018; Tamilselvam et al., 2019).

Typically, ML models are developed using text-based programming languages that require coding, which entails an understanding of the programming concepts and its syntax (McCracken et al., 2001). Therefore, to popularize ML, it is desirable to reduce the cognitive effort so the user can focus on the logic to solve the problem at hand (Knuth & Pardo, 1980). For this purpose, visual languages have been introduced that let users create programs by simply drag-and-drop a visual element on a canvas and subsequently connecting that element with other elements rather than by specifying them textually (Idrees et al., 2018; Weintrop & Wilensky, 2017). Such visual representations can take diverse forms, including block-based or flow-based languages (Burnett & Baker, 1994; Pasternak et al., 2017). Visual languages can improve learnability for novices by helping them to prevent errors, favor recognition over recall, and provide domain-specific limited instruction sets reducing the cognitive load (Çakiroğlu et al. 2018). These advantages have led to widespread adoption within introductory programming contexts across different educational stages (Bau et al., 2017). Especially in K-12, block-based programming languages such as Scratch, SNAP!, Blockly, and App Inventor are widely used for teaching algorithms and programming concepts (Weintrop, 2019).

Following this success, visual tools are also being proposed for teaching ML. These tools typically include a component for the development of an ML model and a deployment component (Rodríguez-García et al., 2020). The ML development component supports collecting and labeling data, building a model using available ML algorithms (learning), evaluating the performance of the model with test data, and exporting the model to a programming platform. On the other hand, the deployment component is needed to develop an application using the ML model created by the ML development component to allow students to create usable intelligent solutions to make computing education engaging.

Yet, so far there are no systematic overviews on visual tools for teaching ML in K-12 and their characteristics. Rodríguez-García et al. (2020) present a comparison of a few tools, whereas Hauck et al. (2019) focus only on tools to develop Internet of Things and AI-based business ideas. Other reviews on visual languages in K-12 focus on teaching computational thinking, not covering ML (Hubwieser et al., 2015; Kraleva et al., 2019; Noone & Mooney, 2018; Weintrop & Wilensky, 2017). Reviews on teaching ML in K-12, such as Marques et al. (2020) provide an overview of existing educational units, without analyzing in detail the adopted tools, and Long and Magerko (2020) focus on the definition of AI/ML literacy. On the other hand, reviews on ML tools in general, such as by Dudley and Kristensson (2018) analyze only the user interface design of interactive ML tools. Therefore, we present in this article the results of a systematic mapping study on visual tools for teaching ML in K-12 of the last decade (2010–2020). The results of this study can help instructional

designers and educators to choose the most appropriate tool as well as researchers to guide the evolution and improvement of these tools.

2 Machine learning education in K-12

Although there have been some AI teaching initiatives in K-12 already in the 1970s (Kahn, 1977; Papert & Solomon, 1971), and involving ML in the 1990s (Bemley, 1999), only recently it has become a trend again (Marques et al., 2020). In this context, the AI for K-12 Working Group (AI4K12) aims at developing guidelines for teaching K-12 students about Artificial Intelligence. To frame these guidelines, "big ideas" in AI that every student should know are defined, including perception, representation & reasoning, learning, natural interaction, and societal impact (Touretzky et al., 2019a). Thus, while AI is "the science and engineering of making intelligent machines that have the ability to achieve goals as humans do", Machine Learning (ML) is a subfield of AI dealing with the field of study that gives computers the ability to learn without being explicitly programmed (Mitchell, 1997). ML algorithms build a mathematical model based on sample data, denoted as "training data", to make predictions or decisions without being explicitly programmed to perform the task. ML can be applied for a wide range of application domains and tasks, including image recognition, object detection, and segmentation, motion and pose recognition, as well as text, sound and speech recognition, sentiment analysis, among others (Blott et al., 2019).

Regarding Machine Learning, the primary goal in K-12 education is to promote students' understanding of how ML works and its limitations, ethical concerns, and societal impacts. Therefore, ML concepts to be covered in K-12 education should include (Touretzky et al., 2019a):

- What is learning and approaches to ML (e.g., regression algorithms, instance-based algorithms, support vector machines, decision tree algorithms, Bayesian algorithms, clustering algorithms, artificial neural network algorithms) as well as types of learning algorithms (i.e., supervised, unsupervised, reinforcement learning).
- Fundamentals, types of neural networks, including also Deep Learning, a subset of neural networks that makes computational multi-layer neural networks feasible, including, e.g., convolutional neural networks (CNNs), as well as types of neural network architectures and how learning is influenced.
- Limitations, concerns, and impact of machine learning.

To achieve the learning of ML competencies on an application level, this requires students to learn how to develop ML applications for them to become creators of intelligent solutions (Kahn et al., 2018; Kandlhofer et al., 2016; Long & Magerko, 2020; Sulmont et al., 2019; Touretzky et al., 2019b). Building such a custom ML application in a human-centric manner is an iterative process that requires students to execute a sequence of steps as presented in Table 1 (Amazon, 2019; Amershi et al., 2019; Mathewson, 2019; Watanabe et al., 2019).

Table 1 Human-centric ML process

Phase	Description
Requirements analysis	During this stage, the main objective of the model and its target features are specified. This also includes the characterization of the inputs and expected outputs, specifying the problem. This may also involve design thinking approaches to define the objectives with existing needs and problems
Data management	During data collection, available datasets are identified and/or data is collected. This may include the selection of available datasets (e.g., ImageNet), as well as specialized ones for transfer learning. The data is prepared by validating, cleaning, and preprocessing the data. Data sets may be labeled for supervised learning. The data set is typically split into a training set to train the model, a validation set to select the best candidate from all models, and a test set to perform an unbiased performance evaluation of the chosen model on unseen data
Feature engineering	Using domain knowledge of the data, features are created including feature transformation, feature generation, selecting features from large pools of features among others
Model learning	Then a model is built or more typically chosen from well-known models that have been proven effective in comparable problems or domains by feeding the features/data to the learning algorithm. Defining network architectures involves setting fine-grained details such as activation functions and the types of layers as well as the overall architecture of the network. Defining training routines involves setting the learning rate schedules, the learning rules, the loss function, regularization techniques, and hyperparameter optimization to improve performance
Model evaluation	The quality of the model is evaluated to test the model providing a better approximation of how the model will perform in the real world, e.g., by analyzing the correspondence between the results of the model and human opinion. The evaluation of ML models is not trivial, and many methods can be applied for model evaluation with various metrics such as accuracy, precision, recall, F1, mean absolute error, among others, which appropriateness depends on the specific task
Model deployment and monitoring	During the production/deployment phase, the model is deployed into a production environment to create a usable system and apply it to new incoming events in real-time

Yet, as machine learning is a complex knowledge area, students may have difficulties with the first steps when learning ML (Sulmont et al., 2019). And, as K-12 students often do not have any prior computing experiences, it is important to carefully define the sequence of learning goals to be achieved. Thus, an effective way to learn ML should begin with lower-level competencies first, then progressing upwards. On the other hand, it is also important to not remain on lower levels, as this may hinder the development of creative competencies requiring open-ended and ill-defined learning activities. Therefore, the “Use-Modify-Create" cycle (Lee & Chen, 2015; Lytle, 2019) commonly used for the progression of learning computing concepts and practices, can also be adopted for ML education. Following this cycle, students ease into ML topics

by first “using" and analyzing a given ML artifact, then “remixing/modifying" an existing one, until eventually “creating" new ones. This progression incorporates a smooth transition from reusing a predefined artifact to learner-generated creative construction. This is important to go beyond coding or using ML applications following predefined tutorials, as these will not provide enough opportunity for a deeper understanding and creativity (Bellettini, 2014). Furthermore, adopting a “computational action” strategy (Tissenbaum et al., 2019), which allows students to learn ML concepts while creating meaningful artifacts that have a direct impact on their lives and their communities, is crucial to give learners the opportunity to be creative and express themselves through the application of ML (Kahn et al., 2020).

In order to support such learning by creating ML models, age-appropriated tools are required that should have a low floor and high ceiling to make it easy for novices to get started and possible to work on increasingly sophisticated projects (Resnick & Silverman, 2005). In addition, they should support and suggest a wide range of different ML models, e. g, ranging from the recognition of pet images to music understanding to allow students to work on projects motivated by their interests and passions (Resnick & Silverman, 2005).

3 Definition and execution of the systematic review

In order to provide an overview of the state of the art on visual tools for the development of custom ML applications in the context of K-12 education, we performed a systematic mapping study following the procedure defined by Petersen et al. (2008).

3.1 Definition of the review protocol

The objective of this study is to answer the research question: What visual tools exist for teaching ML in K-12 through the development of custom ML models? The goal of this work is to characterize and compare these tools, to provide an overview to guide their systematic selection as well as to identify potential gaps and opportunities for future research. Therefore, we analyze the following questions:

AQ1. What visual tools for teaching ML exist?
AQ2. What are their educational characteristics?
AQ3. What are their characteristics concerning the ML platform?
AQ4. What are their characteristics concerning the deployment platform?
AQ5. How have the tools been developed and evaluated?

Inclusion and exclusion criteria. We consider only English-language publications that present a visual tool for the development of ML models, not including generic visual programming languages or tools for other domains. Due to the emergent nature of the topic of the review and the rapid evolution of ML recently, we focus on tools from the last decade (2010–2020). We focus on tools that allow to create custom ML models, excluding tools for demonstration purposes. We also exclude any approach focusing only on the visualization of ML models or aiming at the complete automation of their development. Furthermore, we only include tools that have been developed

or used for educational purposes in K-12. Consequently, we exclude any ML tool targeted exclusively for professional or adult end-users. We consider only articles that present substantial information allowing the extraction of relevant information regarding the analysis questions. Therefore, abstract-only or one-page articles are excluded.

Sources. We searched the main digital databases and libraries in the field of computing, including ACM Digital Library, ERIC, IEEE Xplore Digital Library, ScienceDirect, Scopus, Web of Science, and Wiley with access via the Capes Portal.[1] We also searched on Google to find tools that have not been published in scientific libraries, as it is considered acceptable as an additional source aiming at the minimization of the risk of omission especially regarding tools that may not yet have been published via the scientific databases (Piasecki et al., 2018). In order to further minimize the risk of omission, we also included literature found via backward and forward snowballing (Wohlin, 2014). Secondary literature has been consulted to complete the information on the encountered tools.

Definition of the search string. Based on the research question, several informal searches were performed to calibrate the search string, identifying relevant search terms (Table 2). We also included synonyms to minimize the risk of omitting relevant works. We did not include terms related to education as this in test searches returned mostly articles related to the application of ML techniques for learning analytics or personalized learning, rather than being related to teaching ML concepts. In order to minimize the risk of omission, we searched for the search terms not only in the titles but in the abstracts of the publications.

3.2 Execution of the search

The search was executed in February 2021 by the authors. The initial search retrieved a total of 1,974 artifacts on the scientific bases and 484,000 artifacts on Google (Table 3). Due to the large number of results of some searches, we restricted the analysis to the 300 most relevant ones. We quickly analyzed the search results based on their title and abstract. Irrelevant and duplicate papers returned by multiple searches were removed. This stage left us with 56 potentially relevant artifacts. During the second stage of selection, we analyzed the full text applying the inclusion and exclusion criteria to identify relevant ones.

Focusing specifically on ML, we excluded any tool providing general support for teaching AI such as Logo (Kahn, 1977). Aiming at support for the development of custom ML models, we also excluded tools for demonstration and visualization only, such as Tensorflow Playground (Smilkov et al., 2017), TensorBoard (Wongsuphasawat et al., 2018), and DeepGraph (Hu et al., 2018). We also excluded environments that support well-defined and controlled exercises, such as code.org's AI for Oceans activity,[2] Zhorai (Lin et al., 2020), PopBots (Williams et al., 2019), or Conversational AI

[1] A web portal providing access to scientific knowledge worldwide, managed by the Brazilian Ministry on Education for authorized institutions, including universities, government agencies and private companies (www.periodicos.capes.gov.br).

[2] https://code.org/oceans

Table 2 Search string for each source

Source	Search string
ACM Digital Library	[Abstract: "visual programming"] OR [Abstract: "block-based programming"] OR [Abstract: "gui tool"] OR [Abstract: toolkit]] AND [[All: "machine learning"] OR [All: "neural network"]] AND [Publication Date: (01/01/2010 TO 12/31/2020)]
ERIC	((abstract:“visual programming” OR abstract:“block-based programming” OR abstract:“gui tool” OR abstract:“toolkit”) AND (abstract:"machine learning" OR abstract:"neural network")) pubyear: since 2010
IEEE Xplore Digital Library	(("Abstract":“visual programming” OR "Abstract":“block-based programming” OR "Abstract":“gui tool” OR "Abstract":“toolkit”) AND ("Abstract":"machine learning" OR "Abstract":"neural network")) Filters Applied: 2010—2020
Science Direct	Year: 2010–2020 Title, abstract, keywords: (("visual programming" OR "block-based programming" OR "gui tool" OR toolkit) AND ("machine learning" OR "neural network"))
Scopus	TITLE-ABS-KEY ((("visual programming" OR "block-based programming" OR "gui tool" OR toolkit) AND ("machine learning" OR "neural network"))) AND PUBYEAR > 2009 AND PUBYEAR < 2021 AND (LIMIT-TO (SUBJAREA, "COMP"))
Web of Science	(AB = (("visual programming" OR "block-based programming" OR "gui tool" OR toolkit) AND ("machine learning" OR "neural network"))) AND LANGUAGE: (English) Timespan: 2010–2020. Indexes: SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH, ESCI
Wiley	"visual programming" OR "block-based programming" OR "gui tool" OR toolkit" in Abstract and "machine learning" OR "neural network" in Abstract (Filter 2010–2020)
Google	"block-based" "machine learning"

Table 3 Number of artifacts identified per stage of selection

Source	No. of search results	No. of analyzed artifacts	No. of potentially relevant artifacts	No. of relevant artifacts
ACM	263	263	12	3
ERIC	160	160	2	0
IEEE	310	300	9	2
Science Direct	76	76	3	0
SCOPUS	703	300	8	4
Web of Science	434	300	8	3
Wiley	28	28	0	0
Google	484,000	300	14	5
Backward snowballing			15	6
Forward snowballing			7	3
Total				24 (without duplicates)

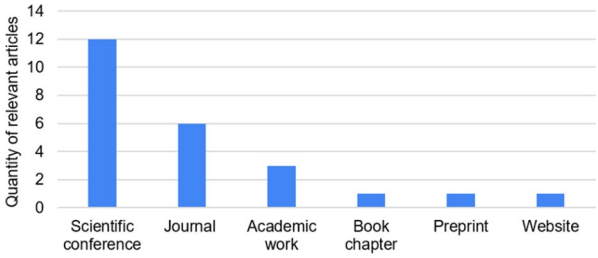

Fig. 1 Quantity of relevant artifacts per type

(Van Brummelen et al., 2019), guiding the development of one specific model (or parts) and/or their deployment only. We also excluded visual tools targeting professional use and for which no applications in K-12 education have been reported, such as Apple Machine Learning,[3] KNIME,[4] Microsoft Azure,[5] Nvidia Digits,[6] Sony Neural Network Console[7] (Hauck et al., 2019; Xie et al., 2019), among others. We also excluded block-based extensions such as BlockPy (Bart et al., 2017) or Jigsaw[8] providing a combination of block-based interfaces with Python in Jupyter notebooks, as they are rather targeted at higher education students. Applying backward and forward snowballing based on the primary studies, we identified 9 additional artifacts. As a result, a total of 24 relevant artifacts of diverse types (Fig. 1) have been identified, representing 16 tools.

4 Results

According to the analysis questions, relevant data has been extracted from the articles. If available, we have also consulted secondary literature, for example, academic works as well as exploring the tools themselves. Data extraction was done independently by the authors and then revised by all authors until consensus was obtained. Varying terminology referring to the same concept has been unified and aggregated.

4.1 What visual tools for teaching ML exist?

As a result, we identified 16 visual tools developed or being used for teaching the development of custom ML models in K-12 (Table 4).

Most of the tools comprehensively support both the development of ML models and their deployment as part of software artifacts, such as games or mobile applications. The deployment is integrated into block-based programming environments

[3] https://developer.apple.com/machine-learning/create-ml
[4] https://www.knime.com
[5] https://azure.microsoft.com
[6] https://developer.nvidia.com/digits
[7] https://dl.sony.com
[8] https://github.com/Calysto/metakernel/blob/3d79efe57175336801169a31d148914c8fb4b0d2/examples/Jigsaw%20in%20IPython.ipynb

Table 4 Visual tools for teaching ML in K-12

Name	Brief description	Scope		Reference(s)
		ML platform	Deployment platform[9]	
AlpacaML	An iOS application that supports users in building, testing, evaluating, and using ML models of gestures based on data from wearable sensors	x	Scratch	(Zimmermann-Niefield et al., 2020; Zimmermann-Niefield et al., 2019a, b)
BlockWiSARD	A visual programming environment that makes use of the WiSARD WANN to enable people to develop systems with some learning capability	x	BlockWiSARD	(Queiroz et al., 2020)
Cognimates	An AI education platform for programming and customizing the development of AI models embodied in devices, such as Amazon's smart speaker Alexa, Cozmo, etc	x	Scratch	(Druga, 2018; Druga et al., 2019)
DeepScratch	A programming language extension to Scratch that provides elements to facilitate building and learning about deep learning models by either training a neural network based on built-in datasets or using pre-trained deep learning models	x	Scratch	(Alturayeif et al., 2020)
eCraft2learn	Additional blocks to the visual programming language Snap! that provides an easy-to-use interface to both AI cloud services and deep learning functionality	x	Snap!	(Kahn & Winters, 2017, 2018; Kahn et al., 2018, 2020)
Educational Approach to ML with Mobile Applications	A set of App Inventor extensions spanning several ML subfields, among which the Teachable Machine extension allows to develop an ML model	x	App Inventor	(Zhu, 2019)

Table 4 (continued)

Name	Brief description	Scope		Reference(s)
		ML platform	Deployment platform[9]	
Google Teachable Machine (TM)	A web-based interface that allows people to train their own ML classification models, without coding, using their webcam, images, or sound	x	–	(Carney, 2020)
LearningML	A platform aimed at learning supervised ML for teaching ML in K-12	x	Scratch	(Rodríguez-García et al., 2020)
mblock	A block and code-based programming software and its Teachable Machine extension that allows to create an ML model	x	mblock	https://www.mblock.cc
Milo	A web-based visual programming environment for Data Science Education	x	–	(Rao et al., 2018)
ML4K	A tool that introduces ML by providing hands-on experiences for training ML systems and building things with them	x	Scratch, App Inventor, Python	(Lane, 2018)
Orange	A data visualization, ML, and data mining toolkit that features a visual programming front-end for exploratory data analysis and interactive data visualization	x	–	(Demšar, 2013; Godec et al., 2019)
Personal Image Classifier (PIC)	A web system where users can train, test, and analyze personalized image classification models with an extension for MIT App Inventor that allows using the models in apps	x	App Inventor	(Tang et al., 2019a, b)
RapidMiner	Comprehensive data science platform with visual workflow design and full automation of ML solutions	x	–	(Sakulkueakulsuk et al., 2018)
Scratch-NodesML	A system enabling children to create personalized gesture recognizers and share them	x	Scratch	(Agassi et al., 2019)

Table 4 (continued)

Name	Brief description	Scope		Reference(s)
		ML platform	Deployment platform[9]	
SnAIp	A framework that enables constructionist learning of Reinforcement learning with Snap!	x	Snap!	(Jatzlau et al., 2019)

[9]We consider the availability of a deployment platform only when the tool allows the model deployment in platforms that can be directly accessed by the students.

typically used in K-12 such as Scratch, App Inventor, and Snap! (Fig. 2). Fewer tools support the integration with text-based languages such as Python.

Most of the tools are available online for free, but some require user registration and/or the use of API keys, which may be confusing for the target audience to obtain and to use (Table 8). The majority of the tools is available in English only. Only mblock and ML4K are available in several different languages supporting a wider application, as typically native languages are required on this educational stage.

4.2 What are their educational characteristics?

Following a strategy of learning by doing the tools are typically used in short duration extracurricular units, either as standalone units of about 1–4 h. In some cases, they are adopted as part of a more comprehensive ML course over several weeks covering also to a larger extend more theoretical knowledge and/or discussing ethical and societal impact of ML as well as career opportunities. And, although there are educational units focusing on younger children (from age 6 up), most concentrate on the high school level, with a considerable number also for middle school level. Several tools do not further specify an educational stage, targeting K-12 in general (Fig. 3).

In accordance with the current generalized lack of knowledge on ML, all tools are primarily targeting novices in ML. However, providing an advanced mode, some

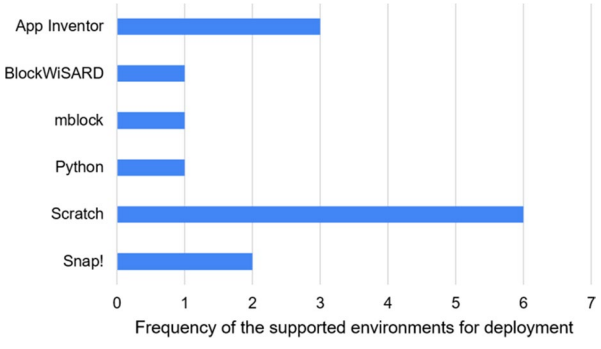

Fig. 2 Supported environments for the deployment of the created ML models

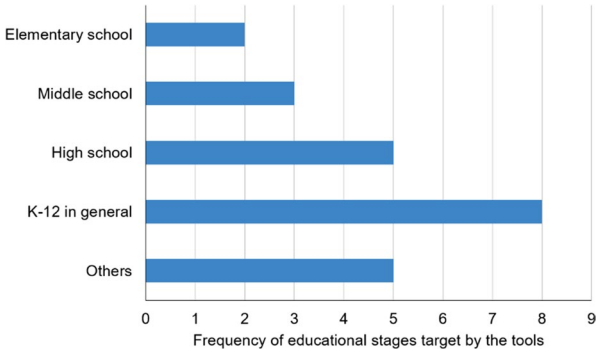

Fig. 3 Frequency of educational stages targeted by the tools

of the tools also enable more knowledgeable users to interact on a more detailed level when building, training, and/or evaluating the ML model. Several tools assume that the users have experiences with the respective programming environment for deployment beforehand, for example, through coding classes.

Most tools are accompanied by educational units that are either just a practical activity typically guided by a step-by-step tutorial and example artifacts or, in some cases, also include a more theoretical part in form of expository lectures and/or videos. Several follow the use-modify-create cycle, encouraging the students also to create their custom ML model in the final stage of the educational unit (Table 5). Adopting a computational action strategy, several courses include the deployment of the developed ML models, allowing students to create a working intelligent solution in the form of a mobile app or game. Applying ML concepts through the creation of ML models, these units also provoke critical analysis of the obtained performance results as well as the strengths and weaknesses of ML in general. Varying degrees of educational support accompany the tools, including mainly step-by-step tutorials for hands-on activities using the tools. Further educational materials include lesson plans, slides, videos, examples, and exercises. Most of the accompanying educational units are available in English only.

4.3 What are their characteristics concerning the ML platform?

We identified three types of tool support for the development of ML models as illustrated in Figs. 4 and 6 and detailed in Table 9. Six tools, such as DeepScratch, eCraft2Learn, ScratchNodesML, and SnAIp provide block-based support by extending the respective programming environment providing specific ML blocks for data preparation, training, and evaluation. Yet, the majority (8 tools), including Google Teachable machine, PIC, and LearningML, adopt a workflow-based approach supporting the development of the ML by guiding the user step-by-step via a web browser or app-based visual user interface. We also encountered reports of the usage of two data flow-based tools (Orange and RapidMiner) for teaching ML in K-12. Adopting a dataflow-based approach, they use boxes as entities, connected by arrows, lines, or arcs which represent directed relations and the data flow. Such data

Table 5 Educational characteristics of the tools

Name	Target audience	Accompanying educational units		
		Type and duration	Educational strategy	Educational resources
AlpacaML	Middle- and high-school students without ML experience	3-h workshop	The tool is demonstrated and students learn to build a predefined model by following an interactive tutorial. Then students build models of their own physical activity by collecting, labeling data, and evaluating the model	–
BlockWiSARD	People in general, including children	–	–	–
Cognimates	Children (7 to 12 years)	1.5–2 h workshops	Children are asked to draw and imagine the future of AI agents. Then they are introduced to different AI agents (Alexa home assistant, Jibo andCozmo robots) by playing with them. Then they get to program the agents with their dedicated coding applications	Lesson plans, example projects, tutorials, teacher guides
DeepScratch	Kids and high-school students	–	–	–
eCraft2learn	Non-experts, K-12 students	–	Students first discuss examples of AI applications and are introduced to Snap! and how to use its AI blocks. Then they experiment with speech synthesis and create programs using image recognition	Learning process, tutorials, and exercises, example programs, videos

Table 5 (continued)

Name	Target audience	Accompanying educational units		
		Type and duration	Educational strategy	Educational resources
Educational Approach to ML with Mobile Applications	High-school students	6-weeks course	A simple curriculum using the extensions following a series of tutorials. Each class is split up into two parts: lecture (up to the first half of the class) and the mobile application development	Lesson outlines, interactive and video tutorials, slides
Google TM	Novices without ML or coding expertise (children and adults)	45 min-4hours class as part of an AI course	Several educational units running from online tutorials instruction step-by-step how to build an ML model to their integration as part of AI courses such as (Payne, 2019), in which students first learn basic concepts of AI/ML and then after a demonstration of Google TM, build a model	Tutorials, example programs, slides, videos
LearningML	Children or people interested in AI	–	–	Video tutorials, user manual (in Spanish only)
mblock	Children (10–13 years)	2 h class	After an introduction of basic concepts, students are guided to build a recycling recognition system	Lesson plans, videos, tutorials
Milo	High-school to undergraduate students in non-computer science fields without programming experience	–	–	–

Table 5 (continued)

Name	Target audience	Accompanying educational units		
		Type and duration	Educational strategy	Educational resources
ML4K	Young people (6 – 19+) (Beginner to advanced level)	1–4 h workshops	Students follow a step-by-step tutorial	Tutorials, teacher guides
Orange	K-12, university students, professionals	1-h to 1-week workshops	Students follow a step-by-step tutorial to solve a particular problem	Lecture notes, video tutorials, example workflows
PIC	High-school students	Workshop with two 50 min classes	The first classes start with a short introduction to basic ML concepts, then students use the PIC to build an ML model. In the second class, students use the extension with the trained models to create intelligent mobile applications	Lesson plan, teacher guide, slides, tutorials
RapidMiner	Professionals, Middle school students	3 days workshop	The first phase introduces ML, and students develop an ML model that predicts the sweetness of mangoes using only the outer physical properties. In the second phase, students have to classify mangoes into different categories. In the third phase, the students aim at using the prediction from ML	–
ScratchNodesML	Children	–	–	–

Table 5 (continued)

Name	Target audience	Accompanying educational units		
		Type and duration	Educational strategy	Educational resources
SnAIp	High-school students	–	Introduction to reinforcement learning, practical activity following a pre-defined tutorial including the deployment, evaluation, and optimization of the model	Tutorials, solution examples, cards, teacher instructions (in German only)

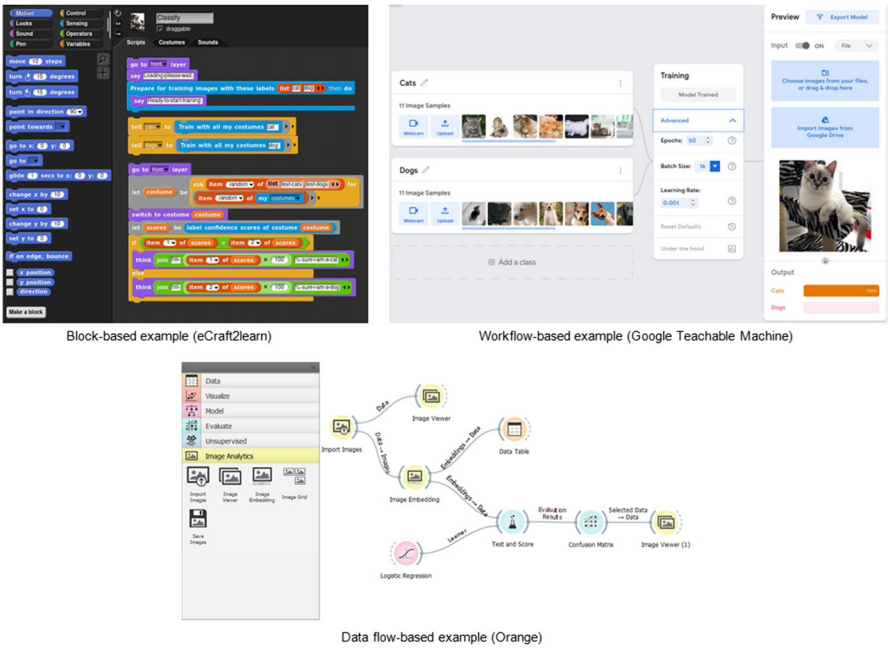

Fig. 4 Examples of ML development support

flow-based tools provide much more detailed and complex functionality, enabling the user to even build the neural network architecture. This approach is typically adopted in visual ML tools for professional use.

All tools are limited concerning the ML task they support, focusing mainly on recognition, such as image or speech recognition, being the tasks in which current ML applications are being very successful (Fig. 5 and Table 9). Only eCraft2learn recently also added blocks supporting object detection and segmentation, while Zhu (2019) supports object detection. Other tasks covered, include speech synthesis by eCraft2Learn, clustering by Milo, and a game agent by SnAIp.

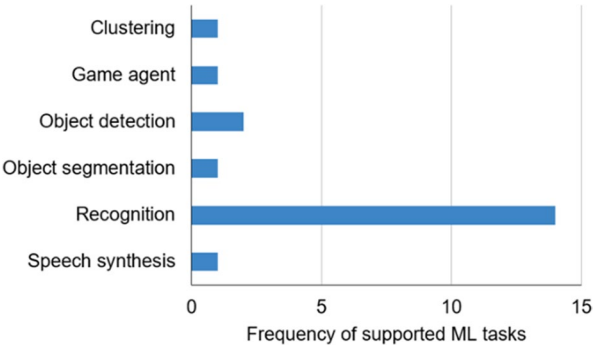

Fig. 5 Frequency of supported ML tasks

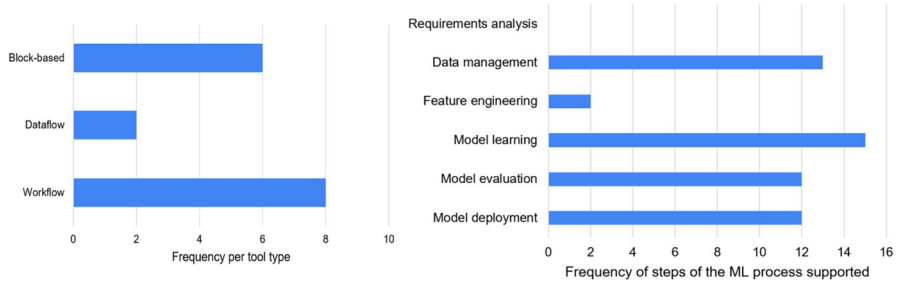

Fig. 6 Frequency of tool types and steps of the ML process supported

The tools largely support all basic steps of a human-centric ML development process (Fig. 6 and Table 9): first, they encourage the collection of small amounts of data and its labeling by organizing it into categories created by the user. Then, this data is used to train an ML model using transfer learning, which allows to build accurate models in a time-saving way, using diverse ML backends such as IBM Watson, Keras, Tensorflow. Once the model is trained, its performance can be evaluated. This is done mostly by allowing the user to test the model with new data for which the model gives as output the predicted label. Fewer tools visualize also performance metrics, such as accuracy and loss function (Carney et al., 2020), or provide support to analyze the performance per image (Godec et al., 2019; Tang et al., 2019). Most tools support also the export of the created model for its deployment either directly to a block-based programming environment or in diverse formats (such as Tensorflow.js, Python, etc.). Yet, none of the tools supports requirements analysis. Feature engineering is also covered only by Orange and RapidMiner, as they support more comprehensively a variety of ML techniques, including decision trees, etc.

In this respect, the tools support an interactive way that allows the students to make any necessary corrections in an informed and systematic manner. The comprehensive support for the complete ML process including the opportunity for the students to perform data preparation and evaluation also enables them to construct a more accurate understanding.

The tools support a variety of data types, mostly images as summarized in Fig. 7 and detailed in Table 10. All of the tools expect users to collect their own data, having students creatively engage with data by incorporating datasets that learners can easily relate to and understand as suggested by Hautea et al. (2017). Data collection is enabled via webcam, microphone, etc. Yet, for example, the collection of even a small sample of images via webcam requires having the related objects nearby and can be somewhat tiresome after a while. Other tools focusing on specific types of inputs enable users to collect data from wearable sensors, or other physical devices, such as Alexa. Several tools also allow uploading files directly from the computer either as a batch or individually, which might be less efficient.

On the other hand, as it may be impracticable for students to collect data during the activities or in order to assure certain characteristics of the datasets, for example, low-dimensional datasets when initially introducing concepts or datasets that

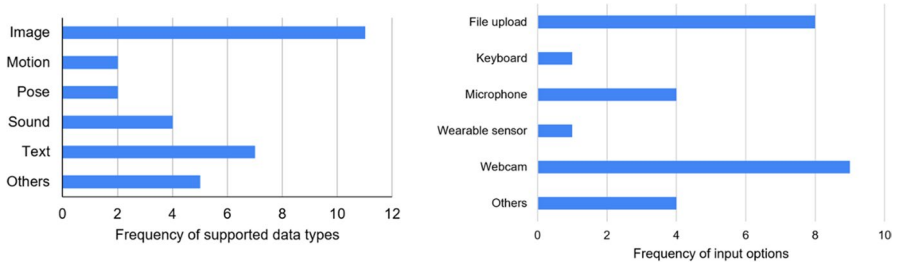

Fig. 7 Frequency of supported data types and input options

are on purpose "messy" when demonstrating issues of bias, it would be helpful to have some example datasets available (D'Ignazio, 2017; Sulmont et al., 2019). However, only a few tools provide easy access to datasets. Google Teachable Machine, for example as part of the "Ethics of Artificial Intelligence Curriculum for Middle School Students" (Payne, 2019) provides easy access to initial datasets available via google drive. Orange provides a widget to directly load data from 65 popular datasets, mainly from the UCI ML Repository and Milo and DeepScratch, the well-known Iris dataset. DeepScratch also provides access to the MNIST dataset containing binary images of handwritten digits. A strength of several tools is the ease with which the dataset can be visually explored via the tools' interface, allowing the user to systematically analyze and, if necessary, adjust the dataset.

As ML Backend, the majority of the tools use common ML frameworks or providers such as Tensorflow or IBM Watson, or their proprietary implementations (Table 6). The predominance of the use of Tensorflow.js can be explained by its ease of execution, without the need for client-side installation or a dedicated infrastructure for the tool. To accelerate training, some tools adopt transfer learning approaches using MobileNet or SqueezeNet as pre-trained deep learning models for image recognition, etc.

In general, the tools support supervised learning, with few exceptions supporting reinforcement learning (Cognimates, ML4K, and SnAIp) and/or unsupervised learning (Orange, RapidMiner). Model training can be performed on the local machine (BlockWiSARD, RapidMiner), with some tools allowing the use of a cloud server (eCraft2learn, Cognimates) or directly on a mobile device (Zhu, 2019). Yet, most use the user's web browser to train the model (Teachable Machine, PIC, LearningML, mBlock). As the model training process can sometimes be slow, tools that allow training locally can make use of the local machine's GPU, when available, to speed up the training process (eCraft2Learn, RapidMiner).

Using visual tools, ML concepts are typically concealed with black boxes to reduce the cognitive load when learning (Resnick et al., 2000). Such abstractions of ML concepts include very high-level representations, as, in ML4K, training the model is reduced to a single action button. Yet, as this concealing of ML concepts limits people's ability to construct a basic understanding of ML concepts (Hitron et al., 2019; Resnick et al., 2000), some tools provide advanced modes that provide a lower-level representation. For example, DeepScratch, eCraft2Learn, Milo and PIC,

Table 6 Characteristics concerning ML model and learning

Name	ML algorithms/backend	Model parameters	Types of learning	Training parameters
AlpacaML	DTW algorithm	–	Supervised	–
BlockWiSARD	WiSARD	–	Supervised	–
Cognimates	IBM Watson SDK and API for custom classifiers, uClassify/ Clarifai	–	Supervised, reinforcement	–
DeepScratch	Dense, RNN, and CNN models or pre-trained models offered byTensorflow.js	Quantity of layers	Supervised	Epochs, batch size
eCraft2learn	Pretrained cloud models, built-in browser support, IBM Watson	Model creation defining layers/ neurons, optimization method, loss function	Supervised	Training iterations, learning rate, validation split, data shuffle
Educational Approach to ML with Mobile Applications	Tensorflow.js	–	Supervised	–
Google TM	Tensorflow.js	–	Supervised	Epochs, batch size, learning rate
LearningML	Tensorflow.js	–	Supervised	–
mblock	NI	–	Supervised	–
Milo	Tensorflow.js	Number of features, type of layers connections, number of nodes, activation function, optimizer function	Supervised and unsupervised	Learning rate, loss function, training metrics, iterations
ML4K	IBM Watson	–	Supervised and reinforcement learning	–
Orange	Diverse ML algorithms, including naive Bayesian classifier, k-nearest neighbors, induction of rules and trees, support vector machines, neural networks, …	Diverse parameters depending on the respective ML algorithm	Supervised and unsupervised	Learning rate, Epochs, Batch size among others

Table 6 (continued)

Name	ML algorithms/backend	Model parameters	Types of learning	Training parameters
PIC	Tensorflow.js	Model (MobileNet or Squeezenet), model type (convolution, flatten), and quantity of layers	Supervised	Learning rate, epochs, training data fraction, optimizer
RapidMiner	Hundreds of ML algorithms	Activation function, hidden layers number, and size, gradient descent method	Supervised, unsupervised	Epochs, number of training data rows to be processed per iteration, learning rate, learning rate decay, momentum, loss function, …
ScratchNodesML	1NN-DTW algorithm	–	Supervised	–
SnAIp	Q-Learning algorithm	–	Reinforcement learning	Learning rate, discount factor, exploration rate, available actions

allow defining parameters of the neural network architecture (such as type of model, number of layers, etc.), while data flow-based tools such as Orange, even provide low-level functionalities to build a neural network from neurons and layers. Such an advanced mode is also provided concerning training parameters (such as epochs, learning rate, batches, etc.) as part of DeepScratch, eCraft2Learn, Google TM, Milo, Orange, PIC, RapidMiner, and SnAIP. Yet, although some tools provide brief information on the vocabulary and/or these parameters, no further help tailored to the target audience is provided to guide the selection of their values.

As uncertainty is an inevitable characteristic of ML models in most real-world applications, and, thus, when interacting with an ML model, it is important that users are aware of this uncertainty to manage their expectations on the model's performance (Dudley & Kristensson, 2018). Yet, the concept of a probabilistic model and its limitations can be difficult to convey to the students, who may have difficulties comprehending the implications as studies show that even a single outlier in a classifier can result in significant confusion for users (Kim et al., 2015). In this respect, most tools also provide support for the evaluation of the trained ML model (Table 11), mostly by allowing the user to test the model with new data (captured via webcam, etc.) for which the model gives as output the predicted label to which the input belongs, and the confidence value representing the hit probability.

Fewer tools visualize also performance metrics, such as accuracy and loss function (Carney et al., 2020) (Fig. 8). Another approach is the visualization of a correctness table and/or confidence graph (Tang et al., 2019) (Fig. 9). A correctness table shows all of the testing images and whether or not they were classified correctly. This helps users to infer why specific images were classified correctly or not by comparing the images to find similarities. A confidence graph shows all testing images for one label at a time based on model confidence. It allows users to infer the characteristics of images that a model learns for specific labels so that users can find

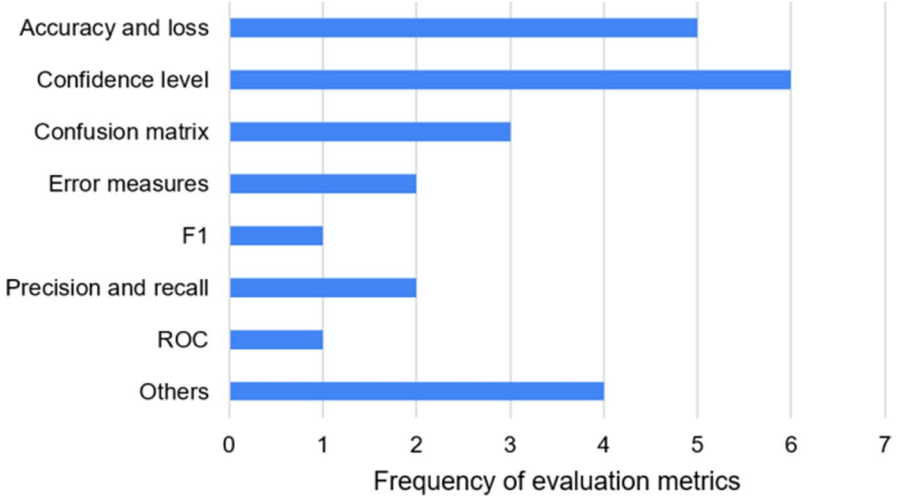

Fig. 8 Frequency of evaluation metrics

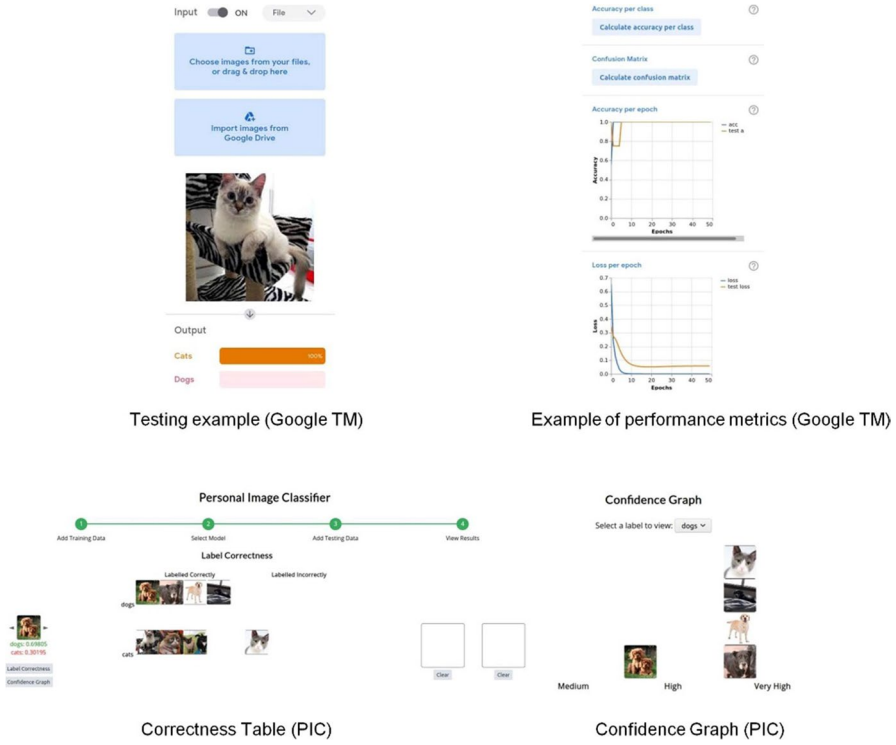

Fig. 9 Examples of support for the evaluation of the ML models

patterns in how the model makes decisions to improve its accuracy. LearningML intends to show in advanced mode also a confusion matrix, a table that in each row presents the examples in a predicted class while each column represents the examples in an actual class. These visualizations of the results of the classification, facilitate the identification of data that are not accurately classified, and thus, support the analysis of the students to improve the model's performance. The use of examples to support the understanding of classes appears to be a promising solution that resonates with users (Kim et al., 2015). Only tools targeting professional use such as Orange and RapidMiner, provide a more complete set of commonly used performance metrics, including mean Average Precision, F1, among others.

Yet, considering the need for understanding of certain mathematical concepts, such as percentages, which are typically taught only at the end of primary school or beginning of middle school, the appropriate application of these concepts depending on the educational stage has to be carefully selected. No further guidance on the interpretation of value ranges and the performance level they indicate is given, neither tips on how to adjust the model if desired performance levels are not achieved. Providing such information as part of an educational tool could help the students to interpret and understand the results and to constructively guide them to learn how to improve the model.

4.4 What are their characteristics concerning the deployment platform?

While some tools just support the export of the created ML model, several provide also support for the deployment as part of a game or mobile application, integrated or as an extension of a block-based programming environment (Fig. 10).

Depending on the specific task(s) the tool supports, ML programming blocks are provided to embed the created ML into the project (Table 12). By far the most adopted block-based environment is Scratch followed by App Inventor (Fig. 2). To embed the created ML models, these extensions provide additional programming blocks. Depending on the variety of tasks supported by the tool, this may range from very few blocks (such as 3 image recognition blocks) to larger sets with up to 119 blocks (DeepScratch) for diverse purposes (Table 12). In general, these new blocks are designed in conformance with the visual design of the respective block-based programming language.

In this way, the tools allow students to learn ML concepts while empowering them to create meaningful artifacts with a direct impact on their lives and their communities. This may motivate them to create innovative applications that match their interests and passions providing a learning tool with wide walls (Kahn et al., 2020).

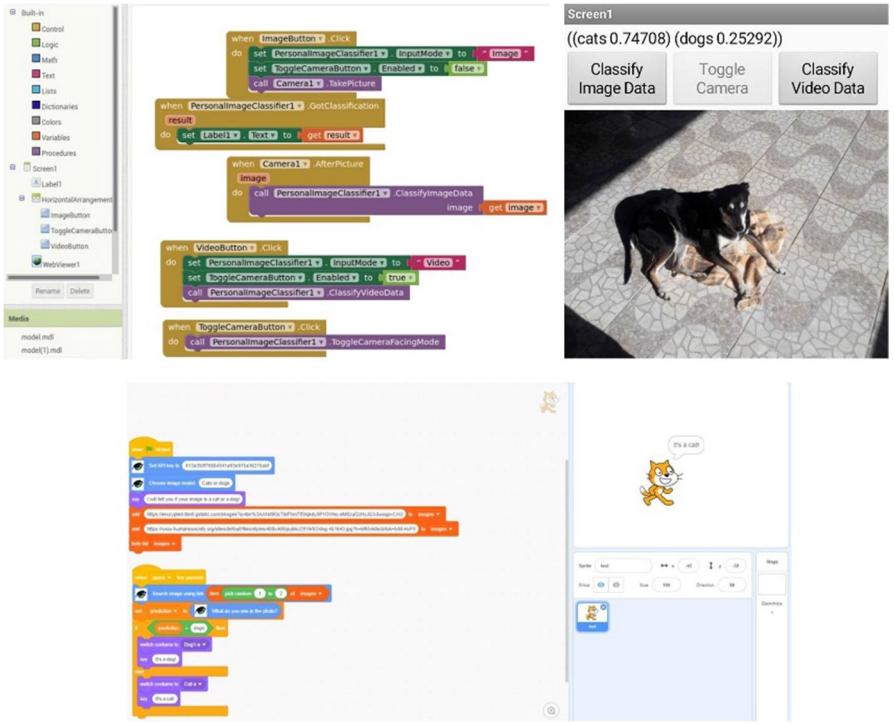

Fig. 10 Example of block-based deployment support (PIC and Cognimates)

4.5 How have the ML tools been developed and evaluated?

Most of the encountered publications lack a description of the research methodology adopted to develop the ML tools (Table 13). Only Queiroz et al. (2020) indicate an alignment with Constructivism, Constructionism and knowledge building and intelligent agents, learning process, and the perception of intelligence, Druga et al. (2019) used a participative design approach with codelab sprites, and Alturayeif et al. (2020) used an incremental agile model. The source code of several tools (Table 13) is available under open-source licenses allowing their evolution and adaptation.

However, several studies aiming at the evaluation of the tools are reported (Table 7). The factors evaluated range from the tools' effectiveness on the students' learning, usability, usefulness, and efficiency to the identification of their strengths and weaknesses. The evaluations have been conducted as case studies or report applications in an informal way (Carney et al., 2020). Sample sizes are mostly small, ranging from 5 to 23 participants, only Kahn and Winters (2018) presented a study with 40 students, Sakulkueakulsuk et al. (2018) with 84 students and Druga et al. (2019) present a large replicated international study including 102 children. The findings of these studies indicate that the tools help to leverage the students' domain knowledge to collect data, build models, test, and evaluate models by allowing them to conduct rapid iterations to test hypotheses about model performance and reformulate their models. They also rated the tools' usability as very good. Furthermore, the tools seem to help students to develop and discuss theories about how the models work, and the characteristics of a good model, thus, helping them to grasp even complex ML concepts as well as to critically reflect on the impact of ML in practice. The integration into general block-based programming environments also enables the creation of engaging and purposeful artifacts.

5 Discussion

Considering the recentness of the trend to teach Machine Learning in K-12, we identified a considerable amount of 16 tools. These tools support exploration allowing students to try out different alternatives and create their custom ML models. Providing a visual interface, the tools allow the students to interact and execute a human-centric ML process in an interactive way using a train-feedback-correct cycle, enabling them to iteratively evaluate the current state of the model and take appropriate actions to improve it. This plays a critical role in order to demonstrate the limitations and implications of ML in practice. Most of these tools have a simple and appealing design that allows intuitive user interaction and easy task accomplishment. The majority of the tools are proposed in the context of block-based programming environments such as Scratch, Snap!, and App Inventor, typically adopted at this educational level.

Although there can be observed a slightly larger number of tools targeting the high-school level, a considerable number of tools also aims at K-12 in general, providing, thus, support for various educational stages. Taking into account the current situation, in which most students in K-12 do not have any previous knowledge on Artificial Intelligence or Machine Learning, most of these tools are appropriately designed for novices. Thus, following the design principles proposed by Resnick

Table 7 Information regarding the evaluation of the tools

Name	Evaluated quality factors	Research design	Sample size	Application context	Findings
AlpacaML	How young people could use AlpacaML to build, test, and refine models of athletic skills that they wish to improve	Case study	6	Students aged 8–14 years who had experience with scratch	Leverages the students' domain knowledge to collect data, build models, test and evaluate models; allows them to conduct rapid iterations to test hypotheses about model performance and reformulate their models; allows students to develop theories about how the model works, and the characteristics of a good model
BlockWiSARD	–	–	–	–	–
Cognimates	Impact of tool design and actors not directly related with the tool (e.g., how do children imagine AI in the future, how is their cultural and social-economical context influence their perception of smart technologies, what role do parents and technology access play)	Series of case studies	102	Children (7–12 years old from schools from 4 different countries	The children developed a rich grasp of AI concepts through play and coding with our platform. They also became more skeptical of the agents'smarts and truthfulness even if they continued to perceive them as friendly and exciting
DeepScratch	Usability in terms of effectiveness, efficiency, and satisfaction	Case study	5	Children	All users completed the two tasks successfully. The usability test indicates that DeepScratch is efficient in supporting users in achieving their goals and tasks in minimal time. In terms of satisfaction, all participants reported that the applications were very easy to create, and supplemented their understanding of deep learning models. However, three students raised the need of translating the blocks into the Arabic Language

Table 7 (continued)

Name	Evaluated quality factors	Research design	Sample size	Application context	Findings
eCraft2learn	Student understanding of AI and agent environment, perception and action, as well as student attention, engagement, and enjoyment and usability of the Snap! environment	Case-study	40	Senior High Schools and Vocational students	77.5% of the students indicated that they understand AI. All students enjoyed the learning process except for one student. More than 40% of the students stated that it was easy. 82.5% of the students were interested and motivated to make the AI program by using Snap!
Educational Approach to ML with Mobile Applications	ML learning and interest	Pre-/post-test case study	10	High-school students	The extensions proved beneficial to grasp the concepts introduced in the first part of the class
Google TM	ML learning	Informal	NI	Middle-school students	Results indicate that Google TM has been useful to introduce ML concepts, allowing an understanding of concepts like bias and fairness. Google TM also seems to facilitate active learning of AI concepts by letting students interact with those concepts by making models themselves
LearningML	Learning of AI knowledge	Pre/post-test case study	14	Adult students with prior programming experience but no previous training on AI	The results of the intervention seem very promising, especially taking into account that this effect was experienced after just 2 h of treatment, which highlights the impact that LearningML had on the learners and draws attention to its potential as a tool to foster AI and ML knowledge
mblock	–	–	–	–	–
Milo	Usefulness and ease of use of the tool, along with the perceived level of understanding of ML concepts	Pre/post-test case study	20	Undergraduate students from a first introductory course in ML	90% of participants reported that visualizations were very easy to create using Milo and supplemented their understanding of the concepts. 70% of students felt the tool would be very useful for novice learners
ML4K	–	–	–	–	–

Table 7 (continued)

Name	Evaluated quality factors	Research design	Sample size	Application context	Findings
Orange	–	–	–	–	–
Personal Image Classifier	Usability of the PIC tools and their effectiveness in introducing novices to ML	Pre-/post-test case study	23	Workshops with high school students	The students enjoyed using the PIC interface and the Expressions Match app, indicating that the tools were intuitive and fun to use. Students were able to use the analysis tools to develop reasoning for how their models were behaving. The way the tool provides visual representations of data-enabled guided discussions about dataset imbalance and how that can lead to what appears to be a biased model
RapidMiner	Learning of ML concepts, fun/engagement, awareness/attitude towards ML	Pre-/post-test case study	84	Middle school students	Based on the results, ML can be used as a powerful tool to successfully conduct interdisciplinary education at the middle school level. Students combined knowledge, observations, and teamwork efforts to achieve the goal of using the ML model for prediction. Students had more fun, engagement, and hands-on interactivity in the workshop compared to their regular classroom, even though the topic of AI is much more complex and challenging
ScratchNodesML	–	–	–	–	–
SnAIp	–	–	–	–	–

et al. (2005), these tools provide a low threshold, high ceiling, and wide walls. By concentrating on essential features allowing users to learn how to use them and to assemble models quickly in a completely no-code fashion, especially workflow-based tools are well aligned with the novices' cognitive process. On the other hand, block-based tools adopting a programming approach to the development of ML models are more complex, requiring the students to have a prior understanding of coding concepts and practices. Data flow-based tools, such as Orange or RapidMiner, that even enable the user to assemble the ML model from scratch, are mostly used in advanced contexts being designed rather for professional use than K-12 education. And, by providing support for the development of custom ML models, the tools also provide support for a "high ceiling", which means that the tools are sufficiently powerful to create sophisticated, complete solutions.

Furthermore, to allow students to create their own ML models adopting a computational action strategy to make computing education engaging, it is also necessary that these tools provide a structure for common processes while remaining flexible to account for variability in problems (Patel, 2010). Considering object recognition being currently one of the most typical ML tasks, the majority of the tools only support the development of ML models for recognition tasks, mostly image recognition. Being one of the most straightforward ML tasks, this seems to be adequate for starting to learn ML. Following the current trend on transfer learning, most of the tools provide adequate support for this technique adopting prominent Deep Learning models, such as TensorFlow.

Following the strategy to make it easy to get started but providing room to use more complex concepts, these tools tend to abstract key operations during the learning process. Especially workflow-based tools are in alignment with this requirement abstracting the ML process as much as possible. For example, training an ML model in ML4K is reduced to click one action button. These tools are also designed most straightforwardly, limiting the features they offer to only essential ones, leaving part of the process as a black box. Yet, this concealing of ML concepts limits people's ability to construct a basic understanding of ML concepts and seems to not only result in a smaller learning effect but no learning at all (Hitron et al., 2019). Therefore, the goal has to be to create an ML learning environment with sufficient scaffolds for novices to start to create ML models with little or no formal instruction (low threshold) while also being able to support sophisticated programs (high ceiling). To simultaneously target different kinds of users, some of the tools (i.e., DeepScratch, Google TM, Orange, PIC, SnAIp) offer advanced modes in which they allow more advanced students to define hyperparameters for training (such as learning rate, epochs, batch size, etc.) or more detailed evaluation metrics while hiding these details from novices. Yet, designing these tools, one of the most important decisions is the choice of the "primitive elements" that users will manipulate. Future research is necessary to explore this issue and identify the balance between uncovering carefully selected underlying concepts while minimizing the cognitive load as much as possible and adequately support learning progress.

Some tools provide more varied support for diverse ML tasks and/or data types (such as images, speech, pose, etc.), which may evolve in the future with K-12 students becoming more advanced in ML. Yet, even providing support for the

development of custom ML models for one specific task already opens an enormous opportunity for exploration and the development of original solutions, contributing also to the development of creativity as an important 21st-century skill. It also provides the opportunity of an interdisciplinary insertion of ML education in traditional K-12 disciplines as well as supporting education paradigms such as "computational action" that aims at learning the creation of useful computational artifacts for social aims, empowering young people to impact their communities by becoming creators in the digital economy (Weintrop et al., 2020). It can be expected that future research will further explore the vast possibilities of applying ML for different data types and tasks such as object detection and natural language processing including support for a larger variety of tasks and data types by these visual tools.

Several tools also provide support for the deployment of the custom ML model as part of a game or mobile application, either integrated or as an extension of a block-based programming environment. Such support is essential to teach not only the development, but also the deployment and consequent usage of the created models, and, thus, demonstrate the usefulness of this knowledge as part of the students' life and community for solving real problems. Yet, regarding some tools, little or no support for deployment is provided. For example, although Google Teachable Machine provides code snippets for deploying the created model as a tensorflow.js within a web browser, more detailed and age-adequate educational support teaching deployment on this educational level is not yet available. Especially considering the popularity of Google Teachable Machine as a visual tool for creating ML models, it would be important to create such support for deployment, including technical infrastructure as well as educational material covering this step.

In accordance with the current situation, in which Artificial Intelligence/ML concepts are not included in K-12 curriculums in most countries, the majority of the educational units developed for teaching ML using the visual tools are designed for short-duration extracurricular activities. Several provide step-by-step tutorials for hands-on activities, as well as lesson plans, slides, etc. Yet, most of the educational resources are available in English only, hindering a wider worldwide adoption, as these materials need to be available in the native language of the students at this educational stage. This indicates not only the need for the translation of the existing educational units to different languages but also the customization of the topics addressed to motivate the students by presenting problems that are of interest in their specific local context, such as e.g., the example of classifying the ripeness of mangos as proposed in the application in Thailand by Sakulkueakulsuk et al. (2018).

Considering these visual tools as a means for learning, we observed that so far, they seem not to provide a more comprehensive educational support, as typically offered by other visual environments aiming at teaching algorithms and programming. This includes for example the provision of instructions and constructive hints as part of the tool environment. And, although some of the block-based visual tools are directly integrated within the same environment in which the deployment of the custom ML models takes place, this is not the case for most other tools. Therefore, a better integration of the visual tools within the deployment environment would eliminate the need for the usage of different tools. Another issue is collaborative learning, commonly

adopted in computing education in K-12 by students working together in small groups to maximize their own and each other's learning and motivation (Bellanca et al., 2010; Gillies, 2016; Johnson & Johnson, 2014). Yet, we did not encounter any support for developing the ML solutions as a team, which may complicate their application in an educational context in practice. Observing also the popularity of sharing projects as part of several block-based programming communities (such as Scratch), only mblock, as part of their community platform, supports students to share their created ML models and/or to remix ones from the community.

Another issue is the lack of support for a performance-based assessment of the created ML models, representing the learning outcome. Neither the tools nor the associated educational units provide any kind of embedded or associated support. Yet, assessment in the learning process is important to provide feedback to the learner, teacher, and other interested stakeholders. And, observing the availability of such assessments, even in automated ways for "traditional" block-based programming languages (e.g., Dr. Scratch (Moreno-León & Robles, 2015) or CodeMaster (Alves et al., 2020; Gresse von Wangenheim, 2018; Solecki et al., 2020), the need for such support also in the context of ML education becomes evident. We also observed that most tools do not provide any kind of support for teachers to monitor their students' learning. An exception is the ML4K tool environment, for which just very recently has been added a teacher supervision functionality that lists the projects and the training data of all students in a class.

This indicates diverse research opportunities by extending the tools' functionalities including support for collaborative teamwork and sharing as well as in some cases a more comprehensive embedding in environments covering also deployment. And, in addition, the integration of teacher support, especially visioning the automation of the performance-based assessment of the learning outcomes created by the students allowing also timely feedback to the students themselves to guide their learning process.

Most of the visual tools are free and are available online, making installation unnecessary, but on the other hand requiring a continuous internet connection during class, which may be a problem in some educational contexts. Some tools require user registration and/or the use of keys which can be complicated for the target audience. Model training is mostly done through cloud-based services such as IBM Watson or Google Machine, which makes advanced machine learning capabilities approachable. Few tools (such as mblocks or ML4K) provide hybrid support, offering a textual programming alternative for more advanced use preparing a transition to text-based ML environments using Python.

Analyzing the publications, we observed a generalized lack of information on how the tools have been developed. Yet, a systematic methodology for the design of such tools including the analysis of the educational context, their design as well as the adoption of a systematic software development process is essential to develop tools that satisfy the needs effectively and efficiently. As with few exceptions, empirical studies evaluating the tools are also rather exploratory regarding the quality factors evaluated, research design as well as sample size. As a consequence, there still seems to be lacking evidence on which tools may be best for certain educational stages, educational contexts, etc. to guide their selection in a sound way. Thus, there is a need for more

empirical research analyzing diverse aspects of these visual ML tools to systematically evolve and improve these tools for better support of ML education in K-12.

Threats to validity. In order to minimize threats to the validity of the results of this study, we identified potential threats and applied mitigation strategies. Systematic reviews suffer from the common bias that positive outcomes are more likely to be published than negative ones. However, we do not consider this a critical threat to our research as rather than focusing on the impact of these tools, we aimed to characterize the tools themselves. To mitigate the omission of relevant studies, we carefully constructed the search string to be as inclusive as possible, considering not only core concepts but also synonyms. The risk of excluding relevant primary studies was further mitigated by the use of multiple databases and the inclusion of secondary literature. Threats to study selection and data extraction were mitigated by providing a detailed definition of the inclusion/exclusion criteria. We defined and documented a rigid protocol for the study selection and all authors conducted the selection together, discussing the selection until consensus was achieved.

6 Conclusion

This paper presents a systematic mapping of visual tools for teaching Machine Learning in K-12 in the last ten years (2010–2020). As a result, we identified 16 tools providing a visual interface that allow the students to interact and execute a human-centric ML process in an interactive way. Most of the tools are targeted mainly at beginners at the high school level or K-12 in general. Following design principles proposed by Resnick et al. (2005), these tools provide a low threshold and wide walls concentrating on essential features to allow users to learn how to use them and to assemble results quickly, while some simultaneously target "high ceilings", offering advanced modes that allow the configuration of the ML process. Most tools focus on recognition tasks, providing support for the complete ML process, from data preparation to evaluation using different types of visual representations. The majority of the tools are integrated into common block-based programming languages, allowing for the direct deployment of ML models created as part of intelligent software solutions. Several tools are accompanied by educational units for teaching, yet, most of them are only available in English. Therefore, further educational material is required to facilitate a wider application in other countries considering also locally relevant problems to adequately motivate the usefulness of this kind of knowledge. Furthermore, the tools need to be enhanced in order to support collaborative teamwork and the sharing of the learning outcomes, as well as to provide in some cases a more direct integration with the deployment environment. This also includes the need for support for automatic assessments guiding in a timely fashion the learning process of the students as well as easing the teachers' effort.

In general, we observed a lack of information on how the tools have been developed and evaluated, although, the results of few explorative empirical studies indicate the usability and usefulness of these tools in K-12. Yet, there is still a need for more empirical research analyzing diverse aspects of these visual ML tools to

systematically evolve and improve these tools for better support of ML education in K-12.

As a result, this mapping can help instructional designers and educators to choose the most appropriate tool for their educational goals and contexts. Furthermore, the results of the mapping also point out several implications for future research in this area, including:

- Development of tools for the introduction of Machine Learning at earlier educational stages, such as middle school, to further popularize knowledge on Artificial Intelligence and Machine Learning.
- Large-scale applications and studies of learning progress in K-12 regarding ML concepts and practices to identify the balance between uncovering carefully selected underlying concepts while minimizing the cognitive load as much as possible.
- Enhancement of tools for the support of a greater variety of ML tasks to offer more contents alternatives and facilitate the interdisciplinary integration of teaching ML into existing K-12 curricula.
- Extending the provision of adequate support for deployment as part of some of the tools, such as the popular Google Teachable Machine, including technical infrastructure and educational material.
- Provision of support for different levels of learners concerning their knowledge in Machine Learning and Programming, as once more students have participated in introductory ML courses, a greater need for intermediate and advanced courses will arise.
- Provision of tool support for different learning modes, such as online learning, especially motivated also by the current COVID-19 situation.
- Analysis of learning performance to improve the underlying learning strategies and consequently the respective tool support to systematically help students to learn Machine Learning concepts effectively, efficiently, and in a creative and engaging way.
- Adoption of rigorous scientific methodologies for the development of tools and their functionality, in addition to the conduction of more rigorous studies analyzing more comprehensively and systematically the impact of these tools on ML learning in K-12.

As the results of this mapping also provide a first indication that the adoption of visual tools for teaching ML in K-12 can be beneficial and provide a valuable contribution especially for novices and considering the current importance of the popularization of AI/ML, it also provides a basis for further research in this area to support the teaching of this innovative knowledge area in K-12.

Appendix

Table 8 General characteristics of the tools

Name	Website of running version of the tool	Platform	Usage license	User registration/API key	Language(s)
AlpacaML	–	online/app	–	Key to connect with Scratch	English
BlockWiSARD	–	desktop	Free	–	English, Portuguese
Cognimates	http://cognimates.me/home	online	Free	required	English
DeepScratch	-	online	NI	-	English
eCraft2learn	https://ecraft2learn.github.io/ai	online	Free	required	English
Educational Approach to ML with Mobile Applications	https://appinventor.mit.edu/explore/resources/ai/image-classification-look-extension	online	Free	–	English
Google TM	https://teachablemachine.withgoogle.com	online	Free	–	English, German, Spanish, Japanese
LearningML	https://learningml.org	online	Free	only for cloud storage and sharing	English, Spanish, Catalan, Galician
mblock	https://www.mblock.cc	online/ desktop/ app	Free	only for cloud storage/ sharing	26 languages, including English
Milo	https://miloide.github.io/	online	Free	–	English
ML4K	https://machinelearningforkids.co.uk	online	Free	required for most projects	More than 15 different languages including English
Orange	https://orange.biolab.si	desktop	Free	–	English
PIC	https://classifier.appinventor.mit.edu	online	Free	–	English
RapidMiner	https://rapidminer.com	online/ desktop	Paid	required	English
ScratchNodesML	–	online	–	–	English
SnAIp	https://ddi.cs.fau.de/schule/snaip	online	Free	–	English, German

Table 9 General characteristics concerning the ML platform

Name	Type of tool	Supported ML tasks	Supported parts of the ML process
AlpacaML	Workflow	Motion recognition	Data management, Model learning, Model evaluation, Model deployment
BlockWiSARD	Block-based	Image recognition	Data management, Model learning, Model deployment
Cognimates	Workflow	Image and text recognition	Data management, Model learning, Model evaluation, Model deployment
DeepScratch	Block-based	Image and numerical recognition	Model learning, Model evaluation, Model deployment
eCraft2learn	Block-based	Image and speech recognition, speech synthesis, object detection, and segmentation	Data management, Model learning, Model evaluation, Model deployment
Educational Approach to ML with Mobile Applications	Workflow	Image and text recognition, object detection	Data management, Model learning, Model deployment
Google TM	Workflow	Image, sound, and pose recognition	Data management, Model learning, Model evaluation
LearningML	Workflow	Image, numerical, sound, and text recognition	Data management, Model learning, Model evaluation, Model deployment
mblock	Workflow	Image recognition	Data management, Model learning, Model evaluation, Model deployment
Milo	Block-based	Data clustering	Model learning
ML4K	Workflow	Image, text, number, sounds, and face recognition	Data management, Model learning, Model evaluation, Model deployment
Orange	Dataflow	Image and text recognition	Data management, Feature engineering, Model learning, Model evaluation
PIC	Workflow	Image recognition	Data management, Model learning, Model evaluation, Model deployment
RapidMiner	Dataflow	Data classification and prediction	Data management, Feature engineering, Model learning, Model evaluation
ScratchNodesML	Block-based	Gesture recognition	Data management, Model evaluation, Model deployment
SnAIp	Block-based	Game agent	Model learning, Model deployment

Table 10 Characteristics concerning data

Name	Types of data	Input options	Availability of datasets ready to use
AlpacaML	Motion	Wearable sensor	–
BlockWiSARD	Image	File upload, webcam	–
Cognimates	Image, Text	File upload	–
DeepScratch	Image, Text	Webcam	Iris and MNIST
eCraft2learn	Image, pose, sound	File upload, webcam, microphone,	–
Educational Approach to ML with Mobile Applications	Image, video stream	Webcam	–
Google TM	Image, pose, sound	File upload, webcam, microphone	Initial Teachable Machines datasets (e.g., cat-dog dataset)
LearningML	Image, sound, text	File upload, keyboard, webcam, microphone	–
mblock	Image	Webcam	–
Milo	Numbers, text	File upload	Few popular datasets used in introductory ML courses, like the Iris dataset
ML4K	Image, sound, text	Upload from a browser, webcam, microphone	-
Orange	Image, text, network graph, gene expression, time series, mosaic spectral image, SQL table	File upload	65 popular datasets from UCI ML Repository and other sources
PIC	Image	File upload, webcam	–
RapidMiner	Numbers, Text	Instant connection to diverse data sources	–
ScratchNodesML	Motion	Via Bluetooth from a physical hardware device	–
SnAIp	Game actions	Snap!	–

Table 11 Characteristics concerning evaluation

Name	Evaluation metrics	Dataset splitting
AlpacaML	Testing with new images indicating their label	After model training, students can add new actions the users are making for testing
BlockWiSARD	–	–
Cognimates	–	After model training, students can add new images for testing
DeepScratch	Training loss & accuracy, Testing accuracy, object correctness/Confidence level	As the model is training, the user can observe how the accuracy and the loss values are being optimized after each epoch. Once the training is done, the accuracy of the testing data will be available to the user
eCraft2learn	Training loss, accuracy, duration	Manual splitting by user
Educational Approach to ML with Mobile Applications	Confidence level shown as black boxes under a class	–
Google TM	Training accuracy and loss function, accuracy per class category	Fixed splitting separating 15% as test data
LearningML	Testing with new images indicating their label together with the degree of confidence. In advanced mode, a set of gauge bars are added to the label indicating the numerical probability, expressed as a percentage, as well as a confusion matrix	–
mblock	Testing indicating the label with the highest confidence level	After model training, students can test new images
Milo	Accuracy plot, Loss plot	–
ML4K	–	–
Orange	Classifier metrics: Area under ROC, accuracy, F-1, precision, recall, Specificity, LogLoss, confusion matrix Regression metrics: MSE, RMSE, MAE, R2, CVRMSE Graphical metrics: ROC analysis, Lift curve, calibration plot	Manual splitting by proportion, by the number of instances, as well as apply cross-validation partitions
PIC	Prediction results per image, correctness table, confidence graph	After model training, students can add new images for testing in the same way as for the training step

Table 11 (continued)

Name	Evaluation metrics	Dataset splitting
RapidMiner	Precision and Recall, Root Mean Square Error, Average Absolute Error, Average Relative Error, Squared Correlation	–
ScratchNodesML	–	–
SnAIp	–	–

Table 12 Characteristics concerning the deployment platform

Name	Export only	Deployment support	
		Platform	**Blocks added for ML**
AlpacaML	–	Scratch extension	–
BlockWiSARD	–	BlockWiSARD	5 blocks for model development
Cognimates	–	Scratch extension	12 vision training blocks, 9 text training blocks, 4 sentiment blocks, 11 input/transformation blocks
DeepScratch	–	Scratch extension	119 blocks
eCraft2learn	–	Snap!	Diverse blocks for several ML tasks
Educational Approach to ML with Mobile Applications	–	App Inventor extension	Diverse block extensions
Google TM	tensorflow.js	–	–
LearningML	–	Scratch extension	3 text recognition blocks, 6 image recognition blocks, 5 sound recognition, 3 number recognition blocks, 4 general ML blocks
mblock	–	mblock extension	3 image recognition blocks
Milo	Python code	–	–
ML4K	–	Mostly Scratch, some for App Inventor and Python	–
Orange	Models in Python pickle format	–	–
PIC	–	App Inventor extension PersonalImageClassifier.aix	The PIC extension has three properties in the designer and 11 blocks in the blocks editor
RapidMiner	Python code	RapidMiner proprietary blocks	User can define new blocks with configurable functions
ScratchNodesML	–	Scratch extension	–
SnAIp	–	Snap! extension	Diverse block extensions

Table 13 Information regarding the development of the tools

Name	Scientific methodology	Code available	Code license
AlpacaML	–	–	–
BlockWiSARD	Based on Constructivism, Constructionism and knowledge building and intelligent agents, learning process and the perception of intelligence	–	–
Cognimates	Participative design with codelab sprites	https://github.com/mitmedialab/cognimates-vm	Open-source
DeepScratch	The Incremental Agile Model is used to develop the DeepScratch extension	https://github.com/Noufst/DeepScratch	–
eCraft2learn	–	https://github.com/ecraft2learn/ai/blob/master/ecraft2learn.js	BSD
Educational Approach to ML with Mobile Applications	–	–	–
Google TM	–	https://github.com/googlecreativelab/teachablemachine-community/	Apache License 2.0
LearningML	–	–	GNU General Public
mblock	–	–	–
Milo	–	https://miloide.github.io	Apache 2.0 License
ML4K	–	https://github.com/IBM/taxinomitis/ https://github.com/kylecorry31/ML4K-AI-Extension	Apache-2.0 License
Orange	–	https://github.com/biolab/orange3	GNU GPL 3.0
PIC	–	https://github.com/mit-cml/appinventor-extensions/tree/extension/personal-image-classifier	Apache-2.0 License
RapidMiner	–	–	Proprietary
ScratchNodesML	–	–	–
SnAIp	–	–	–

Authors' contributions Christiane Gresse von Wangenheim: Conceptualization, Methodology, Conduction of the systematic mapping study (definition, search and extraction & analysis), Writing- Original draft preparation. Jean C. R. Hauck: Conduction of the systematic mapping study (definition, search and extraction & analysis), Writing- Reviewing and Editing. Fernando S. Pacheco: Conduction of the systematic mapping study (definition, search and extraction & analysis), Writing- Reviewing and Editing. Matheus F. Bertonceli Bueno: Conduction of the systematic mapping study (extraction & analysis), Writing- Reviewing and Editing.

Funding This work was supported by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico – www.cnpq.br), an entity of the Brazilian government focused on scientific and technological development (Grant No.: 303674/2019–9).

References

Alturayeif, N., Alturaief, N., Alhathloul, Z. (2020). DeepScratch: Scratch Programming Language Extension for Deep Learning Education. *International Journal of Advanced Computer Science and Applications, 11*(7), 642–650. https://doi.org/10.14569/IJACSA.2020.0110777

Alves, N. d. C., Gresse von Wangenheim, C., Hauck, J. C. R., Borgatto, A. F. (2020). A Large-scale Evaluation of a Rubric for the Automatic Assessment of Algorithms and Programming Concepts. In Proceedings of the 51st *ACM Technical Symposium on Computer Science Education,* ACM, 556–562. https://doi.org/10.1145/3328778.3366840

Amazon. (2019). Amazon Machine Learning, AWS Documentation. https://docs.aws.amazon.com/machine-learning/latest/dg/building-machine-learning.html

Agassi, A., Erel, H., Wald, I. Y., & Zuckerman, O. (2019). Scratch Nodes ML: A Playful System for Children to Create Gesture Recognition Classifiers Proceedings of the Conference on *Human Factors in Computing Systems*, ACM 1–6 https://doi.org/10.1145/3290607.3312894

Amershi, S. et al. (2019). Software Engineering for Machine Learning: A Case Study. In Proceedings of the 41st International Conference on Software Engineering: *Software Engineering in Practice, IEEE Press,* 291–300. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Bart, A. C., Tibau, J., Tilevich, E., Shaffer, C. A., & Kafura, D. (2017). BlockPy: An Open Access Data-Science Environment for Introductory Programmers. *Computer, 50*(5), 18–26. https://doi.org/10.1109/MC.2017.132

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: blocks and beyond. *Communications of the ACM, 60*(6), 72–80. https://doi.org/10.1145/3015455

Bellanca, J. A. et al. (2010). 21st Century Skills: Rethinking how Students Learn. Solution Tree Press.

Bellettini, C., et al. (2014). Informatics education in Italian secondary school. *ACM Transactions on Computing Education, 14*(2), 1–5. https://doi.org/10.1145/2602490

Bemley, J. L. (1999). Neural networks for precollege students. *Proceedings of the International Joint Conference on Neural Networks, 6*, 4422-4427.

Blott, M., Halder, L., Leeser, M., Doyle, L. (2019). QuTiBench: Benchmarking Neural Networks on Heterogeneous Hardware. *Journal on Emerging Technologies in Computing Systems, 15*(4), Article 37 https://doi.org/10.1145/3358700

Burnett, M. M., & Baker, M. J. (1994). A Classification System for Visual Programming Languages. *Journal of Visual Languages and Computing, 5*, 287-300.

Çakiroğlu, Ü., Suiçmez, S. S, Kurtoğlu, Y. B., Sari, A., Yildiz, S., Öztürk, M. (2018). Exploring perceived cognitive load in learning programming via Scratch. *Research in Learning Technology, 26.*

Carney, M., et al. (2020). Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification. *Proceedings of Conference on Human Factors in Computing Systems, ACM,*. https://doi.org/10.1145/3334480.3382839

Demšar, J., et al. (2013). Orange: data mining toolbox in Python. *Journal of Machine Learning Research, 14*, 2349-2353.

D'Ignazio, C. (2017). *Creative data literacy. Information Design Journal, 23*(1), 6-18.

Druga, S. (2018). Growing up with AI: Cognimates: from coding to teaching machines. Master thesis, MIT, USA.

Druga, S. Vu, S. T., Likhith, E., Qiu T. (2019). Inclusive AI literacy for kids around the world. *Proceedings of Fab Learning* ACM 104-111. https://doi.org/10.1145/3311890.3311904

Dudley, J. J., Kristensson, P. O. (2018. A Review of User Interface Design for Interactive Machine Learning. ACM Transactions on Interactive Intelligent Systems, 8(2), Article 8. https://doi.org/10.1145/3185517

Forbes. (2019). AI goes to high school. https://www.forbes.com/sites/insights-intelai/2019/05/22/ai-goes-to-high-school/#68826e3f1d0c

Gillies, R. (2016). Cooperative Learning: Review of Research and Practice. *Australian Journal of Teacher Education, 41*(3), 39–51. https://doi.org/10.14221/ajte.2016v41n3.3

Gresse von Wangenheim, C., et al. (2018). CodeMaster – Automatic Assessment and Grading of App Inventor and Snap! *Programs. Informatics in Education, 17*(1), 117–150. https://doi.org/10.15388/infedu.2018.08

Godec, P. et al. (2019). Democratized image analytics by visual programming through integration of deep models and small-scale machine learning. *Nature Communications, 10,* Article 4551. https://doi.org/10.1038/s41467-019-12397-x

Gutosk, M., et al. (2017). Qualitative Analysis of Deep Learning Frameworks. *Journal of the Brazilian Society on Computational Intelligence, 15*(1), 45–52. https://doi.org/10.21528/LNLM-vol15-no1-art3

Hautea, S., Dasgupta, S., Hill, B. M. (2017). Youth perspectives on critical data literacies Proceedings of the Conference on *Human Factors in Computing Systems,* ACM 919–930 https://doi.org/10.1145/3025453.3025823

Hauck, M., Machhamer, R., Czenkusch, L., Gollmer, K., & Dartmann, G. (2019). Node and Block-Based Development Tools for Distributed Systems with AI Applications. *IEEE Access, 7*(143109–143119), 2019. https://doi.org/10.1109/ACCESS.2019.2940113

Hiner, J. (2017). AI will eliminate 1.8M jobs but create 2.3M by 2020, claims Gartner. https://www.techrepublic.com/article/ai-will-eliminate-1-8m-jobs-but-create-2-3m-by-2020-claims-gartner

Hitron, T., Orlev, Y., Wald, I., Shamir, A., Erel, H., Zuckerman, O. (2019). Can Children Understand Machine Learning Concepts? The Effect of Uncovering Black Boxes. In Proceedings of the Conference on *Human Factors in Computing Systems*. ACM, Paper 415, 1–11. https://doi.org/10.1145/3290605.3300645

Hu, Q., Ma, L., Zhao, J. (2018). DeepGraph: A PyCharm Tool for Visualizing and Understanding Deep Learning Models. In Proceedings of the 25th Asia-Pacific Software Engineering Conference, Nara, Japan, 628–632, https://doi.org/10.1109/APSEC.2018.00079

Hubwieser, P., et al. (2015) A Global Snapshot of Computer Science Education in K-12 Schools Proceedings of the ITiCSE on Working Group Reports, Vilnius, Lithuania 65–83. https://doi.org/10.1145/2858796.2858799

Idrees, M., Aslam, F., Shahzad, K., Sarwar, S. M.. (2018). Towards a Universal Framework for Visual Programming Languages. *Pakistan Journal of Engineering and Applied Sciences.*

Jatzlau, S. Michaeli, T., Seegerer, S., Romeike, R (2019). It's not Magic After All – Machine Learning in Snap! using Reinforcement Learning Proceedings of IEEE Blocks and Beyond Workshop, Memphis, TN, USA 37–41 https://doi.org/10.1109/BB48857.2019.8941208

Johnson, D. W., & Johnson, R. T. (2014). Cooperative Learning in 21st Century. *Annals of Psychology, 30*(3), 841–851. https://doi.org/10.6018/analesps.30.3.201241

Kahn, K. M. (1977. Three Interactions between Al and Education. Machine Intelligence, 8.

Kahn, K. M., Winters, N. (2018. AI Programming by Children. In Proceedings of the Conference on Constructionism, Vilnius, Lithuania.

Kahn, K. M., Megasari, R., Piantari, E., Junaeti, E. (2018). AI Programming by Children Using Snap! Block Programming in a Developing Country. In Proceedings of the 13th European Conference on Technology Enhanced Learning, Leeds, UK, 2018.

Kahn, K. M., Lu, Y., Zhang, J., Winters, N., Gao, M. (2020). Deep learning programming by all. In Proceedings of the Conference on Constructionism, Dublin, Ireland.

Kahn, K. M., Winters, N. (2017). Child-Friendly Programming Interfaces to AI Cloud Services. In: Lavoué É. et al. (eds) Data Driven Approaches in Digital Education. Lecture Notes in Computer Science, vol 10474. Springer, Cham. https://doi.org/10.1007/978-3-319-66610-5_64

Kandlhofer, M, Steinbauer, G, Hirschmugl-Gaisch, S., Huber, P (2016). Artificial Intelligence and Computer Science in Education: From Kindergarten to University *Proceedings of IEEE Frontiers in Education Conference*, Erie, PA, USA 1–9 https://doi.org/10.1109/FIE.2016.7757570

Kim, B., Glassman, E., Johnson, B., & Shah, J. (2015). *iBCM: Interactive Bayesian Case Model Empowering Humans via Intuitive Interaction*. Technical Report.

Knuth, D. E., Pardo, L. T. (1980). The early development of programming languages. In A history of computing in the twentieth century, 197–273.

Kraleva, R., Kralev, C., & Kostadinova, D. (2019). A Methodology for the Analysis of Block-Based Programming Languages Appropriate for Children. *Journal of Computing Science and Engineering, 13*(1), 1–10. https://doi.org/10.5626/JCSE.2019.13.1.1

Lane, D. (2018). Explaining Artificial Intelligence. Hello World, 4.

Lwakatare, L.E., Raj, A., Bosch, J. Olsson, H. H., Crnkovic, I. (2019). A Taxonomy of Software Engineering Challenges for Machine Learning Systems: An Empirical Investigation. In: Kruchten P. et al. (eds) Agile Processes in Software Engineering and Extreme Programming. Lecture Notes in Business Information Processing, vol 355. Springer, Cham. https://doi.org/10.1007/978-3-030-19034-7_14

Lee, M. R., & Chen, T. T. (2015). Digital creativity: research themes and framework. *Computers in Human Behavior, 42*, 12–19. https://doi.org/10.1016/j.chb.2014.04.001

Lin, P., Van Brummelen, J., Lukin, G., Williams, R., Braezeal, C. (2020). Zhorai: Designing a Conversational Agent for Children to Explore ML Concepts. In Proceedings of the 10th Symposium on Educational Advances in Artificial Intelligence, New York, NY USA.

Long, D., Magerko, B. (2020). What is AI Literacy? Competencies and Design Considerations. In Proceedings of the Conference on Human Factors in Computing Systems. ACM, 1–16. https://doi.org/10.1145/3313831.3376727

Lytle, N., et al. (2019). Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes Proceedings of the *Conference on Innovation and Technology in Computer Science Education,* ACM 395–401 https://doi.org/10.1145/3304221.3319786

Marques, L. S., Gresse von Wangenheim, C., & Hauck, J. C. R. (2020). Teaching Machine Learning in School: A Systematic Mapping of the State of the Art. *Informatics in Education, 19*(2), 283–321. https://doi.org/10.15388/infedu.2020.14

Mathewson, K. W. (2019). A Human-Centered Approach to Interactive Machine Learning. arXiv:1905.06289v1 [cs.HC].

McCracken, M. et al. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In Working group reports from ITiCSE on Innovation and Technology in Computer Science Education, ACM, 125–180.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Education. New York.

Moreno-León, J., Robles, G. (2015). Dr. Scratch: a web tool to automatically evaluate Scratch projects. In Proceedings of the 10th Workshop in Primary and Secondary Computing Education, London, UK, 132–133. https://doi.org/10.1145/2818314.2818338

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education, 5*(2), 149-174.

Payne, B. H. (2019). An Ethics of Artificial Intelligence Curriculum for Middle School Students. https://docs.google.com/document/d/1e9wx9oBg7CR0s5O7YnYHVmX7H7pnITfoDxNdrSGkp60/view

Papert, S., Solomon, C. (1971). Twenty things to do with a computer. Artificial Intelligence Memo, Number 248, MIT, USA.

Pasternak, E., Fenichel, R., Marshall, AN (2017). Tips for Creating a Block Language with Blockly Proceedings of the IEEE Blocks and Beyond Workshop, Raleigh, NC, USA 21–24 https://doi.org/10.1109/BLOCKS.2017.8120404

Patel, K. (2010). Lowering the barrier to applying machine learning. In Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology, ACM 355–358. https://doi.org/10.1145/1866218.1866222

Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M. (2008). Systematic mapping studies in software engineering. In Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, 68–77.

Piasecki, J., Waligora, M., Dranseika, V. Google Search as an Additional Source in Systematic Reviews. Sci Eng Ethics. 2018;24(2):809–810. https://doi.org/10.1007/s11948-017-0010-4 (2018).

Queiroz, R. L., Sampaio, F. F., Lima, C., Lima, P. M.V. (2020). AI from concrete to abstract: demystifying artificial intelligence to the general public. arXiv:2006.04013 [cs.CY].

Ramos, G., Meek, C., Simard, P., Suh, J., & Ghorashi, S. (2020). Interactive machine teaching: a human-centered approach to building machine learned models. *Human-Computer Interaction*. https://doi.org/10.1080/07370024.2020.1734931

Rao, A., Bihani, A., Nair, M. (2018). Milo: A visual programming environment for Data Science Education Proceedings of the Symposium on Visual Languages and Human-Centric Computing, Lisbon, Portugal 211–215 https://doi.org/10.1109/VLHCC.2018.8506504

Resnick, M., Berg, R., & Eisenberg, M. (2000). Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation. *The Journal of the Learning Sciences, 9*(1), 7–30. https://doi.org/10.1207/s15327809jls0901_3

Resnick, M., Silverman, B. (2005). Some reflections on designing construction kits for kids. In Proceedings of the Conference on Interaction Design and Children. ACM, 117–122. https://doi.org/10.1145/1109540.1109556

Resnick, M. et al. (2005. Design Principles for Tools to Support Creative Thinking. In Proceedings of the Workshop Creativity Support Tools.

Rodríguez-García, J. D., Moreno-León, J., Román-González, M., Robles, G. (2020). LearningML: A Tool to Foster Computational Thinking Skills Through Practical Artificial Intelligence Projects. Distance Education Journal, 20(63). https://doi.org/10.6018/red.410121

Sankaran, A., Panwar, N., Khare, S., Mani, S., Sethi, A., Aralikatte, R., Gantayat, N. (2018). Democratization of Deep Learning Using DARVIZ. *In Proceedings of the AAAI Conference on Artificial Intelligence.*

Sakulkueakulsuk, B. et al. (2018). Kids making AI: Integrating Machine Learning, Gamification, and Social Context in STEM Education. In Proceedings of the Int. Conference on Teaching, Assessment, and Learning for Engineering, Wollongong, Australia, 1005–1010. https://doi.org/10.1109/TALE.2018.8615249

Smilkov, D., Carter, S., Sculley, D.,Viegas, F. B., Wattenberg, M. (2017). Direct-manipulation visualization of deep networks. arXiv:1708.03788 [cs.LG]

Solecki, I. et al. (2020). Automated Assessment of the Visual Design of Android Apps Developed with App Inventor. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, ACM, 51–57. https://doi.org/10.1145/3328778.3366868

Sulmont, E., Patitsas, E., Cooperstock, J. R. (2019). Can you teach me to machine learn? In Proceedings of the 50th ACM *Technical Symposium on Computer Science Education,* ACM, 948 – 954. https://doi.org/10.1145/3287324.3287392

Tang, D., Utsumi, Y., Lao, N. (2019). PIC: A Personal Image Classification Webtool for High School Students. In Proceedings of the IJCAI EduAI Workshop, Macao, China.

Tang, D. (2019). Empowering Novices to Understand and Use Machine Learning With Personalized Image Classification Models, Intuitive Analysis Tools, and MIT App Inventor, M.Eng thesis, MIT, Cambridge, USA.

Tamilselvam, S. G. et al. (2019). A visual programming paradigm for abstract deep learning model development. In Proceedings of the 10th Indian Conference on Human-Computer Interaction. ACM, Article 16, 1–11. https://doi.org/10.1145/3364183.3364202

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM, 62*(3), 34–36. https://doi.org/10.1145/3265747

Torrey, L. (2012). Teaching Problem-Solving in Algorithms and AI. In Proceedings of the 3rd Symposium on Educational Advances in Artificial Intelligence, Toronto, Canada.

Touretzky, D. S. et al. (2019a). K-12 Guidelines for Artificial Intelligence: What Students Should Know. In Proc. of the ISTE Conference, Philadelphia, PA, USA.

Touretzky, D. S. et al. (2019b). Envisioning AI for K-12: What Should Every Child Know about AI? In Proc. of the 33rd AAAI Conference on Artificial Intelligence, Honolulu, HI, USA.

Van Brummelen, J. Shen, J. H., Patton, E. W. (2019). The Popstar, the Poet, and the Grinch: Relating Artificial Intelligence to the Computational Thinking Framework with Block-based Coding. In Proceedings of the Int. Conference on Computational Thinking, Hong Kong, China.

Watanabe, Y. et al. (2019). Preliminary Systematic Literature Review of Machine Learning System Development Process. arXiv:1910.05528 [cs.LG].

Weintrop, D. (2019). Block-based Programming in Computer Science Education. *Communications of the ACM, 62*(8), 22–25. https://doi.org/10.1145/3341221

Weintrop, D., Holbert, N., Tissenbaum, M. (2020). Considering Alternative Endpoints: An Exploration in the Space of Computing Educations. In Proceedings of the Constructionism Conference, Dublin, Ireland, 2020.

Weintrop, D., Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High-School Computer Science Classrooms. *ACM Transactions on Computing Education, 18(1),* Article 3. https://doi.org/10.1145/3089799

Williams, R., Park, H. W., Oh, L., Breazeal, C. (2019). PopBots: Designing an Artificial Intelligence Curriculum for Early Childhood Education. In Proceedings of the 9th Symposium on Educational Advances in Artificial Intelligence, Menlo Park, CA, USA.

Wohlin, C. (2014). 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. ACM, Article 38, 1–10.https://doi.org/10.1145/2601248.2601268

Wollowski, M. et al. (2016). A Survey of Current Practice and Teaching of AI. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA.

Wong, G. K. W., Ma, X., Dillenbourg, P., & Huan, J. (2020). Broadening artificial intelligence education in K-12: where to start? *ACM Inroads, 11*(1), 20–29. https://doi.org/10.1145/3381884

Wongsuphasawat, K. et al. (2018). Visualizing dataflow graphs of deep learning models in tensorflow. IEEE Transactions on visualization and computer graphics, 24(1), 1–12, 2018. https://doi.org/10.1109/TVCG.2017.2744878

Xie, C., Qi, H., Ma, L., Zhao, J. (2019). DeepVisual: A Visual Programming Tool for Deep Learning Systems. In Proceedings of the 27th International Conference on Program Comprehension (ICPC), Montreal, QC, Canada, pp. 130–134, https://doi.org/10.1109/ICPC.2019.00028

Zhu, K. (2019). An Educational Approach to Machine Learning with Mobile Applications. M.Eng thesis, MIT, Cambridge, MA, USA.

Zimmermann-Niefield, A., Polson, S., Moreno, C., Shapiro, R. B. (2020). Youth making machine learning models for gesture-controlled interactive media. *In Proceedings of the Interaction Design and Children Conference.* ACM, 63–74. https://doi.org/10.1145/3392063.3394438

Zimmermann-Niefield, A., Shapiro, R. B., & Kane, S. (2019). Sports and machine learning: How young people can use data from their own bodies to learn about machine learning. *XRDS, 25*(4), 44–49. https://doi.org/10.1145/3331071

Zimmermann-Niefield, A., Turner, M., Murphy, B., Kane, S. K., Shapiro, R. B. (2019b). Youth Learning Machine Learning through Building Models of Athletic Moves. In Proceedings of the 18th *International Conference on Interaction Design and Children. ACM,* 121–132. https://doi.org/10.1145/3311927.3323139