"AlphAI": Teaching AI Algorithms to K12 by Training Learning Robots and Visualizing How It Works

Marie Absalon, Thomas Deneux

Paris-Saclay Institute of Neurosience
National Center for Scientific Research (CNRS) & Paris-Saclay University
151 route de la Rotonde, 91400 Saclay, France
thomas.deneux@cnrs.fr

Abstract

Given the massive transformation of all areas of society by AI, it is becoming essential to integrate AI literacy into the various school curricula from an early age. However, teaching the basic concepts of AI and Machine Learning (e.g. training a model; artificial neural networks) at the K-12 level might seem too abstract, whereas teaching only how to use AI fails to really "open the black box".

To overcome these difficulties, we have developed AlphAI, a software resource designed to make the understanding of AI algorithms accessible and attractive to the general public and children as young as 8 years old. This is achieved by making AI very concrete, first by manipulating the learning of educational robots that users train for different behaviors, such as circuit racing, using either supervised or reinforcement learning; second by visualizing in real time in a graphical interface the details of AI algorithms (neural networks, k-nearest neighbors, Q-learning, etc). In addition, the use of the software is not limited to beginners, since it allows to write one's own AI in Python to control the robots.

In this paper, we present the basic principles of the software, its graphical interface, how to use it with various educational robots, and example activities with classes from Elementary school to University.

AlphAI software and robotic kits are commercially available from Learning Robots.

Introduction

For the past two decades, experts in the fields of Education and Artificial Intelligence have emphasized the importance of imparting quality education in AI from kindergarten through 12th grade (K-12) (Touretzky et al. 2019; Long and Magerko 2020; Wong et al. 2020; Eguchi 2022). Today, these expert recommendations align with a growing public awareness regarding AI's profound impact on society and the job market. Consequently, there is an increasing demand for AI education. However, this demand primarily centers on acquiring skills in using AI tools and harboring concerns about the future of an AI-driven world, rather than gaining a deep understanding of "how AI functions." We argue that teaching not only the practical application of AI but also the underlying AI algorithms represents the most effective approach to demystify AI and empower students.

Various graphic interfaces have been developed to introduce children to machine learning, such as (MachineLearningForKids 2017; Cognimates 2018; TeachableMachine 2017; Vittascience 2021). They share common principles, allowing users to train their own AI for classification tasks. Users select the type of data they want to use (webcam images, sound, text), define categories for classification, gather training data for these categories, then press a button to start training a classification model, test the trained model, and eventually can incorporate it into a block code (e.g. Scratch). These interfaces serve as powerful tools for comprehending the fundamental role of data in AI, the process of training supervised learning models, and studying potential issues like underfitting, overfitting, and biases. Furthermore, they enable users to apply AI within more complex projects (Druga and Ko 2021; Williams et al. 2019). However, these interfaces do not open the "black box" of the AI model itself and leave it perceived as overly complex or even magical. Also, younger pupils will be able to train and test models, for example to recognize objects recorded with a webcam, but might not be able to use these models in fun projects since this requires block coding mastery.

AI algorithms are based on intuitive concepts that can be explained to non-experts, including K-12 students. Numerous popularization videos (e.g., (3Blue1Brown 2017)) and web animations (Karpathy 2014; Tensorflow Playground) allow users to tinker with and grasp neural networks. The goal of AlphAI software is to make such animations accessible to a K-12 audience (and beyond) through a graphical interface that enables students to train neural networks and other AI algorithms using their own data. This bridges the gap between the manipulations offered by existing K-12 AI interfaces and a deeper visualization and understanding of the algorithms.

Moreover, to make AI more tangible, AlphAI focuses on training AI to control robotic devices. Educational robotics has proven effective in promoting computational thinking due to its hands-on nature (Chevalier et al. 2022). It provides students with both practical experience and comprehension, of both supervised learning (AI learns from a dataset with manually-predefined labels) but also of reinforcement learning (AI learns "alone" through trial and error). Reinforce-

ment learning is considered more challenging than supervised learning, and there are fewer initiatives aimed at explaining it to K-12 students (though see (Zhang et al. 2022)). Nonetheless, it presents a powerful approach to tackle the fundamental questions of how machines can learn in full autonomy, and how they compare to humans, questions that lie at the core of people's concerns regarding AI.

Description of the Software Resource

Software Overview

We present a software resource name "AlphAI". This software currently runs on personal computers (Windows, MacOS, ChromeBook, linux; a web-based version is under development). It allows taking control of educational robots (currently: AlphAI robot, a modified version of the camera-equipped Waveshare's AlphaBot robot; Lego Spike; mBot 1; Thymio; Buddy; there will be more in the future), and easily "train" them using either Supervised Learning or Reinforcement Learning.

Figure 1 displays a typical screenshot of the main software interface, once we connected to a robot. Briefly, the software's central view displays the AI input, internal details and output. What is shown for AI internal details depends on some visualization parameters, here the training dataset is displayed. A main control board for training and using the AI model sits on the bottom, while more controls for configuring the robot, AI and other visualization options are available in the configuration tabs on the left.

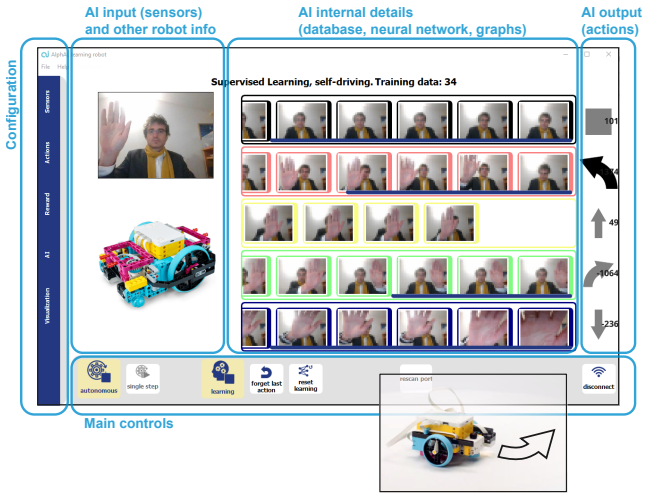

Figure 1: Software overview and photo of controlled Lego Spike robot (bottom inset).

A Tangible Experience of Machine Learning

AlphAI software offers seamless training of a Machine Learning (ML) model taking control of a physical device.

Supervised Learning Supervised Learning is the most common form of Machine Learning:

- User *trains* the AI by remote driving the robot, either by pressing the action icons on the right of the GUI, or using keyboard arrow keys. At this stage button "learning" in the control board should be activated but not button "autonomous". When user hits an action, the software moves the robot and stores its sensor data and the selected action in the dataset. In Fig. 1 example, the sensor is the PC webcam, and the AI has been trained to stop the robot if the user's hand is not visible (top row in the dataset display) and make the robot move in various directions according to hand gestures (rows 2-5).
- Once enough data points have been collected, user hits the "autonomous" button to *use* the AI, letting it drive the robot! Note that user did not need to explicitly start model training as is the case in e.g. Teachable Machine, because model training starts automatically as soon as a first data point is acquired. (It is possible though to uncouple data acquisition and model training when running the program in expert mode.) In the example, user is currently raising his left hand, therefore the AI will drive the Lego car to the left, as it was trained for.

It is thus quick to train the AI by first showing a few examples, then letting it "imitate" (aka "generalize from") these examples and drive the robot. This gives the possibility to kids from 8 years old to enjoy fun robotic activities, such as donkey car races, see use cases below. They will also discover that a good training can be tricky. The training data might include errors, which causes *explicit biases*, or may not cover enough situations, which leads to generalization errors or *implicit biases*. To improve their training, users can curate the dataset by removing incorrect data points or relabelling them (i.e. reassigning their action), and by adding more data points.

Figure 2: Software view with sensor configuration tab opened (left) and screenshot of Thymio robot simulator (bottom inset).

The AI input is not restricted to specific formats such as camera image, but can be any combination of various robot sensors. This will allow performing machine learning activities using the simple sensors of common education robots (infrared, ultrasound, color sensor, etc.), repeating common

activities such as line tracking, object following, obstacle avoidance, but by providing driving example rather than by writing code. As an example Fig. 2 shows a training of Thymio robot using its 7 infrared proximity sensors - 5 in the front and 2 in the back. The choice of which sensor input and which actions to activate is made in the left configuration tabs, or by opening in the menus some predefined configuration for the current robot model. Here Thymio robot was trained to move toward objects activating its sensors (first 6 lines of training data), and stop if no object is detected (last line). As a consequence, once the autonomous button is hit, it will move forward towards the red brick in its front.

Using AI with simple sensor inputs presents numerous advantages: (i) It is possible to visualize and understand all the details of an AI driving the robot based on simple sensors (see next section "Open the black box"), whicih would not be the case for an AI analyzing camera image. (ii) Students will assimilate that the core difference between classis coding and ML is not that the first applies to simple sensors and the second to images and other complex inputs; but it is that ML involves adaptation and learning from data, whereas classic coding consists in the execution of deterministic steps completely foreseen by the programmer. Despite using AI is often irrelevant for analyzing simple sensor data, in some cases it becomes particularly useful and relevant: for example when we use Lego color sensor to recognize objects by their color (RGB values), it would be difficult to write a code to distinguish objects of similar colors (such as red and brown, or red and orange) that works well even when changing perspective and distance, whereas training an AI to do this distinction just by providing examples is straightforward. (iii) Training an AI in our interface is in fact easier than block programming, therefore teachers could consider teaching basic robotic knowledge to young kids - such as sensors, actuators, digitization, decision taking - by first exposing them to an AI interface rather than to a block coding interface! (iv) Finally many schools already have educational robots equipped only with simple sensors, and can do AI activities with them without needing to buy new camera-equipped robots.

Reinforcement Learning and Its Similarities with Human Learning Our software also allows training robots with Reinforcement Learning (RL). In RL, the AI does not need a human-made database of examples of "how to perform" but must find it out by "trial and errors". The AI continuously receive scores called "rewards", which it attempts to maximize by finding the best actions. Reinforcement Learning is popular for training bots playing board games (chess, Go) or computer games, or to train simulated and real robots. Teaching Reinforcement Learning is particularly interesting because it displays particular AI behaviors which are absent in Supervised Learning, in particular *exploration* where the AI tries and evaluates new actions and *anticipation* where actions are chosen for their effect at a later time. There are only few existing attempts to teach Reinforcement Learning to K-12 (Zhang et al. 2022; Martin et al. 2023; Martin, Deneux, and Chevalier 2024).

A simple RL activity that works with all robot models is obstacle avoidance learning: a robot receives maximal reward when moving forward, but negative reward ("penalty") when being blocked by obstacles. Robots will first learn to step backward when hitting obstacles, and later to anticipate and turn before hitting walls. Depending on the used sensors (ultrasound, camera, etc.) learning takes 5 to 150 minutes. For some robot models we also propose balance learning (reach an equilibrium on a balance), soccer learning (rewards are based on approaching a colored ball), line tracking learning (rewards are given only when driving on the line - this learning unfortunately takes 2 hours!).

When running in RL configuration, the software displays two additional indicators: a "reward" bar indicates the reward received after each action, and a "level" bar indicates the average reward received over the last minute. Note that the reward calculation is not part of the AI itself: it is rather an auxiliary program which feeds reward values to the AI while its learns.

When running a Reinforcement Learning activity, the robot learns "all by itself", but students can interfere with its learning by sometimes taking control of the robot, which amounts to forcing its exploration, and thereby accelerate its learning by showing him examples of rewarding actions. We make them realize however that if they take full control of the robot and don't let it make its own errors, the robot won't learn what is not good for him! For example, in the obstacle avoidance task it might learn only that "going forward" is the most rewarding action and continue to run forward even when hitting a wall. Another way of making kids interact during RL activity is to let them tinker the reward or the exploration parameters and observe consequences on learning.

Open the Black Box of AI through Visualization

In addition to the dataset visualization, AlphAI software provides visualizations of AI algorithm details: real-time artificial neural network activity and 1D and 2D "state graphs".

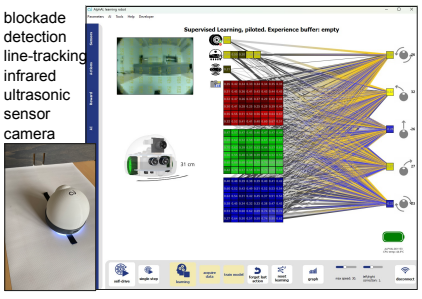

Figure 3: Illustration of the digitization of sensor data and output.

Understand the Digitization of Sensor Data and Actuator Commands The first of the "Five Big Ideas in AI" proposed by the AI4K12 community (Touretzky et al. 2019) is Perception, and more precisely the concept that the external world can be converted to *numbers* by sensors. Indeed the first step before diving into AI algorithms is to play with sensors and visualize the numbers they produce; this is illustrated in Fig. 3 where all sensors of the AlphAI robot have

been activated and appear as input to a small artificial neural network. Users can inspect how blockade detection is digitized to a binary (true/false) value, line-tracking infrared reflection and ultrasound distance to numbers, and camera pixels to RGB values.

Sensor digitization is the preamble for information processing by the AI model. It is important to understand that the model just performs calculations on these numbers and output numbers as well, which are converted into probabilities for actions.

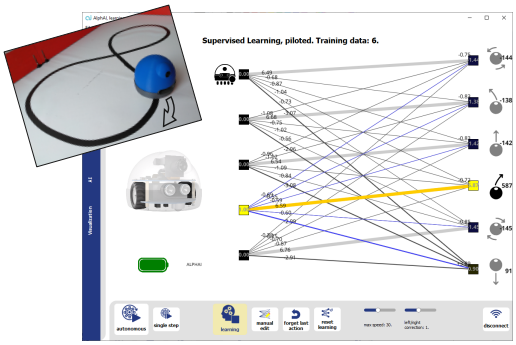

Figure 4: Simple Artificial Neural Network in the context of learning line-tracking.

Artificial Neural Networks Basics The operations of artificial neuron networks (ANN) are generally apprehended as overly complex, because people have in mind ChatGPT or the human brain, and even the usual simple illustrations involve images, which are high-dimensional objects. But educational robots based on simple sensors offer unique opportunities to capture the essence of ANN learning in a much understandable way (similarly to how the 2-sensors 2-nerves tortoise machines of William Grey were a strong inspiration about imitation of life (Walter 1950)). For example Fig. 4 depicts ANN learning of line-tracking behavior using AlphAI robot's 5 line-tracking infrared sensors, here configured as binary black line detectors. Training consisted in driving straight when the line is well-centered on the sensors, turn left when the line drifts to the left, and right when it drifts to the right. Upon training the neural network develops meaningful thick connections: for example on top, a connection between the left-most sensor and the "rotate left" action. Right now, the fourth sensor ("intermediate right sensor") is activated and, as a result of the strong connection of value 6.59, the selected action is "mildly turn to the right".

Note that our visualizations of ANN use different colors for the weights (i.e. connections between neurons; positive weights = light gray, negative = dark gray, absolute values indicated by connection widths), the activity of neurons and how these activity spread through the weights (positive = yellow, negative = blue), and the effect of learning in terms of weights consolidation or decrease (green/red; not appearing in the figure). We also have a manual edition mode that lets users draw their own connections and test them: in this way users "program" themselves the neural network, taking the place of the AI learning algorithm (this is used for example in our "Four levels of autonomy" curriculum below).

Understand the Training Data and Generalization: 1D and 2D Graph, KNN Algorithm The next step in understanding ANN is to understand the nonlinearity of deep learning ANN' hidden layers. To that end we developed some activities where only a single sensor is activated, and the expected output decision is nonlinear: for example emit an alarm if an ultrasonic distance is either smaller or larger than a small range around a reference value. Such output cannot be produced by linear, direct input-output, neural network. This statement might sound abstract, but becomes natural when visualizing the values of output neurons as a function of the single input value, so we provide such visualization called 1D graph (not shown here).

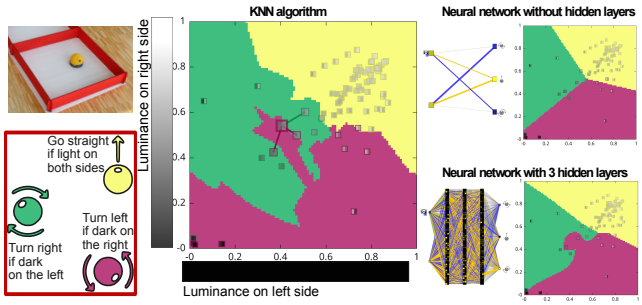

Figure 5: Example of more advanced 2D graph visualization during a 2-pixel camera navigation activity.

We also developed a 2D graph showing the decisions of the model as function of two input values. Such graphs are very common in Machine Learning teaching since they show how a model generalizes from the training data (which appears as a set of points inside the graph) to the full 2D space. Fig. 5 displays an example for a camera navigation activity: The input sensor data consists in the camera luminance in the left and right sides of the robot. A training data has been acquired where robot should turn left when approaching a wall (decreasing luminosity) on its right, and vice-versa, and go straight when image is sufficiently bright. The 2D graph visualizations compare the learnings / generalizations of 3 different AI models: the K Nearest Neighbors (KNN) algorithm, a linear neural network, and a 3-hidden layers neural network. We observe overfitting in the case of KNN and 3-hidden layers neural net, and underfitting in the case of linear network.

Reinforcement Learning: Q-Learning Algorithm Step-by-Step As surprising as it might sound, many details of Reinforcement Learning can be easily explained. To do so, we focus on the Q-learning algorithm, which aims at estimating the values (called "Q-values") of different actions in terms of future rewards expectation, depending on the current state (i.e. sensor values). We designed an activity where the AI learns obstacle avoidance using a single "blockade sensor" allowing only to discriminate between two discrete states blocked / unblocked. The software allows to run the algorithm step-by-step and visualize the learnings resulting from each step. Very briefly: when the AI tries moving for-

ward it learns that this is rewarding (the corresponding connection in the mini 2x5 neural network increases) and repeats it, but when it tries moving backward it learns that this is not rewarding (the connection decreases) and avoids repeating it; we also observe how it learns to anticipate long term reward despite the immediate penalty by reinforcing going backward when being in the blocked state.

Python Coding

The software offers two distinct ways to interact with Python programming.

Student's Own Code for the AI When using the software to study AI algorithms at high-school and University levels, students are expected not only to understand basic algorithms, but also to code them. Therefore among the different choices for supervised and reinforcement learning algorithms in the software, one can choose "Python code" to use one's own code for AI training and decision taking. This allows students to focus on AI coding, while all the other aspects of robot control (communication with PC, sensor data preprocessing, definition of actions, etc.) remains handled by the main software. Students need to code three functions:

- `init(n_input: int, n_output:int)` to initialize their AI model,
- `learn(X_train: np.ndarray, y_train: np.ndarray) -> float` to perform (only) one step of model training - arguments are the training data points (sensor data inputs and associated action labels),
- `take_decision(x: np.ndarray) -> int` to apply the model to a new data input.

As an illustration, here would be the way to write a student code of the K Nearest Neighbors (KNN) algorithm, which is particularly intuitive and easy to code. KNN has the particularity that the model is the training data itself, therefore function `init` will simply initialize an empty global variable to store the training data, and `learn` will simply copy `X_train` and `y_train` to this global variable. Function `take_decision` will contain the actual code of KNN, calculating all the distances between point `x` and the points in `X_train`, determining which of these points are the nearest neighbors, and returning the most represented decision in `y_train` for these neighbors.

Direct Control with Python of the Integrated Robots Professors and students may also be interested to have more direct interactions with the robots. Since we made significant efforts in the AlphAI software for communicating wireless with different models of robot in realtime, to be able to collect their sensors data and send them motor command repeatedly, we made this communication available to users through a Python API that lets users control the robots directly from the PC. Note the advantage, as compared to usual robots' API, that there is no need to upload code and then run the code on the robot.

Pedagogical Resources and Use Cases

Pedagogical Resources

Beyond the presented software, the AlphAI solution is also a pedagogy for teaching AI: a unique way of initiating to Machine Learning that conciliates fun activities and exigence for deep understanding of algorithms. As hinted by the structure of the previous sections, we open the "black box" of AI in 3 steps: (i) *Discover* Machine Learning in a tangible and fun way, (ii) *Visualize* algorithm details to open the black box, and (iii) *Code* algorithms. Naturally the third step is reserved to higher education.

We have developed a number of activities and curricula adapted to different student levels, and to the different integrated robot models, which follow this discover $\rightarrow$ visualize pattern (or discover $\rightarrow$ visualize $\rightarrow$ code for higher education). They can be found on the website of the Learning Robots company, which commercializes AlphAI, in the form of freely available video tutorials, web descriptions and lab pdf (https://learningrobots.ai/en/resources).

These activity resources make connections with mathematical notions studied in secondary and high school. For example, the K nearest algorithm with a 2-value sensor input (Fig. 5 left) relies on computing the distance between points in a 2D graph, and when the number of neighbors K is equal to 1, the separations between regions is made of line segments traced on the perpendicular bisectors of pairs of points belonging to different classes. As an other example, the value of an artificial neuron y linearly connected to a single input neuron x is equal to $y = w * x + b$, where w is the connection weight and b the neuron bias. This is the equation of a straight line, which can be plotted in the software's 1D graph. Our "intruder detection" activity makes student calculate where two such lines corresponding to two output neurons do intersect, before complexifying by adding a hidden layer with an activation function.

The exact mapping between our activities and educational standards in different countries remains yet to be done. Note also that all of the "Five Big Ideas in AI" (Touretzky et al. 2019) can be addressed in our activities. For example, in relation with Big Ideas #4 "Natural interactions" and #5 "Societal impact" we provide a resource for a 3-hours workshop entitled "AI and Ethics", which uses the AlphAI robot to illustrate common ethical problems such as private data collection, biases due to incomplete training data, moral dilemna for an autonomous car, etc.

We are also leading experimentations to assess the impact of AlphAI solution for AI learning, as well as learning in general (Martin et al. 2023; Martin, Deneux, and Chevalier 2024), see some details below. A team at the Education University of Hong-Kong has also created its own curricula using the resource and is currently assessing it in forty grade 5 classes (Kong and Yang 2023, 2024).

Below we present in more details three of our most successful activities and curricula, ranging from general public animation to University course.

Figure 6: Robot race with 5th grade pupils. In the back, screenshot from the software during the activity.

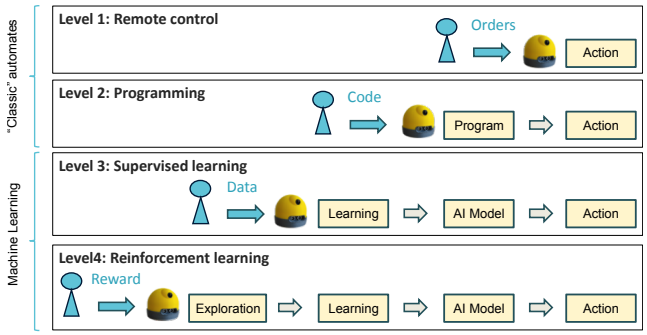

Figure 7: Illustration of the "4 Levels of Autonomy of a Machine".

The Autonomous Robot Race: A Discovery Activity for Everybody (15 Minutes to 1 Hour)

The AlphAI robots race is a very engaging, yet instructive, activity that participants can attend over even short periods of time - as low as 15 minutes - when it is organized during festivals or other events where there is a constant flow of public. It reproduces the concept of Donkey car races, which are popular in engineering schools, and makes them plug-and-play for everybody. We designed a modular arena system to build circuits with enclosing walls, allowing robots to race against each other rather than against time. During about 10 minutes of training time, participants create the training data by remote driving their robot and an AI is immediately trained on this data. During preliminary tests of the AI, competitors assess their performance, and curate and enrich their database to avoid AI driving errors. Then a race is organized between the AI-powered robots.

We have been animating this activity for four years with great success. Its competitive aspect makes it very engaging; it attracts girls as much as boys, the round and colored design of AlphAI robots makes them appear friendly rather than geek-oriented. The activity is also highly instructive as kids understand the importance of data quality. When spending longer time with a group (e.g. 1 hour) we also engage discussion with kids using slides that detail the key learnings from the activity (training a model, "explicit" and "implicit" biases, etc.; see online available material). We also deploy an adaptation of this activity in corporate AI training.

Four Levels of Autonomy of a Machine (2 to 6 Hours)

We have developed a curriculum to teach robotics and AI in a broad sense, entitled "4 Levels of Autonomy of a Machine" and illustrated in Fig. 7. Its aim is to make the distinction between the different ways of configuring an automatic system, and in particular clarify when it involves Machine Learning or not:

1. *Remote Control level* [15 to 45 minutes]: Students control the robots remotely.

   Discussion: The robot is not autonomous since the human chooses its action at every moment.

2. *Programming level* [20 to 45 minutes]: Students program the robots by establishing connections in a mini neural network between possible states of its sensor (seeing or not a wall) and possible actions (go forward, backward, or turn).

   Discussion: The robot now became autonomous. To achieve this it needed to collect and process sensory information (Big Idea #1 and #2). The decision rules were 100% chosen by the human who coded the program.

3. *Supervised (i.e. "Imitation") Learning* [40 to 90 min]: We lead the robot race activity explained above.

   Discussion: The decision rules are now created by the AI itself! The role of the human was to provide examples from which the AI learnt these decision rules. We emphasize how important is this concept of Machine Learning (Big Idea #3) and recapitulate the two stages training/use. We also relate the quality of learning outcome to the quality of the training data and make the distinction between "implicit" and "explicit biases" (see our definitions earlier). We develop on how more generally AI biases can impact society (Big Idea #5).

4. *Reinforcement (i.e. "Trial and Error") Learning* [40 to 90 min]: Students choose how much reward the robot should receive when going straight, and how much penalty when hitting walls. Then they watch the robot learn obstacle avoidance by trial and error. In all cases robots will learn to make straight lines and turn before walls, but depending on the ratio of reward against penalty the learnt behavior will appear more daring (making long straight lines and sometimes still bumping the walls) or more shy (turn a lot in loops).

   Discussion: The human does no longer need to provide data! The AI is making exploration to produce its own data, and learns from this data to adjust its decision rules. Yet the role of the human remains pivotal since it is him who sets the goal of the AI by setting the reward system.

This backbone of 4 manipulations can be deployed with students of very diverse ages: on the one hand the manipulation are sufficiently tangible for kids to grasp the main

concepts from 10 years old, on the other hand these concepts are new for most students and need to be taught at all levels until University.

Assessment on AI and Metacognition Learning in Elementary School (Martin et al. 2023) In Spring 2022 we led an experiment based on these activities in six classes from CE2 to CM2 (grade 3-5, total 138 students), published in (Martin et al. 2023), which we summarize here. The aim of the study was to assess how much could pupils at this level learn about AI concepts, and also how an AI workshop could help them make progress in terms of *metacognition* (Simons, Metzger, and Sonnenschein 2020).

Indeed, questioning with students how an AI robot learns raises questions about how much does this learning resemble human learning by some aspects, and differ by some other aspects. Resemblances are particularly numerous in the case of Reinforcement Learning (RL), as the RL agent needs to explore different strategies, makes error, and needs to repeat explorations and errors before acquiring the expected behavior. Hence during the experimentation, we reproduced the "4 levels of autonomy" workshop detailed above, and when reaching the fourth RL level we followed the recommendations of (Ojeda-Ramirez, Rismanchian, and Doroudi 2023) and emphasized the importance of curiosity, error acceptance, and perseverance in the robot's learnings, and made the students realize how much it is important as well in their own learnings! We also stressed the differences between human and AI: reasoning, sensitivity, and maybe the most important: AI does not set its own goal, it is the human who sets the AI's goal.

The workshop was led in 2 hours. It was preceded and followed by discussions with the students on separate days, to first introduce the concept, and later collect their representations on AI and its uses, making the total time 6 hours. In addition our study involved pre-test and post-test evaluations of students' AI and metacognitive knowledge. Data analysis proved that the workshop led kids to increase both their AI knowledge (e.g. increased from 39% correct answers pretest to 65% post-test to question "Do you think that AI is capable of learning anything that a human brain is capable of learning?") and metacognitive knowledge (e.g. increased from 65% to 81% agreeing to "I think that to learn, it is important to persevere even when the task seems difficult").

High-School or Undergraduate Degree Course (up to 24 Hours)

Since three years, we are giving a course "Initiation to AI" at Paris-Saclay University to first years scientific students who have no background in computer programming and did not necessarily choose computer science as their major. The course consists in 12 sessions of 2 hours detailed in Table 1. Note how it follows both the discover $\rightarrow$ visualize $\rightarrow$ code pattern, as well as the "4 levels" progression (remote control: session 2; programming sensors: session 3; supervised learning: 4-8; reinforcement learning: 9-10).

About 250 students attended this course in three years. They appreciated its concrete aspect with robots and visualization, which helped them to understand intuitively several classic algorithms over a short time period (and also helped them socializing during the first class). To save time the actual robots were used only during sessions 1, 2, 3, 8 and 10; we used the robot simulator during other sessions.

1	Discover AI	AI introductory slides. Practice Supervised Learning with a "Robot racing" competition.
2	Python crash course	Basic programming of controling robot with the keyboard.
3	Python crash course	Program the robot's decision depending on its camera image.
4	Discover & Visualize KNN	Discovering the K nearest neighbors (KNN) algorithm in our graphical interface.
5	Code KNN	Program KNN in python (using "student code" system explained above).
6	Graded test	
7	Discover & Visualize neural nets	("Intruder detection" activity) Manual editing of the weights of a mini-neural network. Discover the need for nonlinearity in a hidden layer.
8	Code neural net	Configure a neural network using scikit-learn python library (using student code system).
9	Discover & Visualize Q-learning	Discovery of Q-learning in the software (the robot learns obstacle avoidance).
10	Code Q-learning	Programming of the algorithm in a particularly simplified case.
11	Opening	Recap slides. Exploration or more advanced AI models. Generative AIs.
12	Graded test	

Table 1: University course plan

Conclusion

We have built a resource, following the assumption that young students can - and in fact should - understand "how does AI work". This will help them, not only to master modern AI as researchers, engineers, or informed end users, but also to participate to the civic debate on how to efficiently use and regulate AI for the greater of society.

We have proved that AI algorithms rely on intuitions that can be explained even to children (at least for the basic concepts of e.g. neural networks), provided we expose them to tangible manipulations and visualizations rather than mathematical formalizations. We also opened the way for new research on how to foster students' metacognition (i.e. reflections on their own learning strategies) by analyzing the requirements for effective self-learnings of an AI.

Further work will include mapping our activities to education standards, and developing toy models of generative AIs to close the gap between small models that can be studied in details and large modern AI constructions.

References

3Blue1Brown. 2017. Neural networks course.

Chevalier, M.; El-Hamamsy, L.; Giang, C.; Bruno, B.; and Mondada, F. 2022. Teachers' Perspective on Fostering Computational Thinking Through Educational Robotics. In Merdan, M.; Lepuschitz, W.; Koppensteiner, G.; Balogh, R.; and Obdržálek, D., eds., *Robotics in Education*, 177–185. Cham: Springer International Publishing. ISBN 978-3-030-82544-7.

Cognimates. 2018. http://cognimates.me/home/.

Druga, S.; and Ko, A. J. 2021. How do children's perceptions of machine intelligence change when training and coding smart programs? In *Interaction design and children*, 49–61.

Eguchi, A. 2022. AI-powered educational robotics as a learning tool to promote artificial intelligence and computer science education. In *International Conference on Robotics in Education (RiE)*, 279–287. Springer.

Karpathy, A. 2014. ConvNetJS: Deep Learning in your browser. https://cs.stanford.edu/people/karpathy/convnetjs/.

Kong, S. C.; and Yang, Y. 2023. Designing and evaluating an attention-engagement-error-reflection (AEER) approach to enhance primary school students' artificial intelligence literacy and learning-to-learn skills: A pilot study. In *Proceedings of the 31st International Conference on Computers in Education, December 4-8, 2023, Matsue, Shimane, Japan*.

Kong, S. C.; and Yang, Y. 2024. Using the robot-assisted attention-engagement-error-feedback-reflection (AEER) pedagogical design to develop machine learning concepts and facilitate reflection on learning-to-learn skills: Evaluation of an empirical study in Hong Kong primary schools. In *Proceedings of the 16th International Conference on Computer Supported Education, May 2-4, 2023, Angers, France*.

Long, D.; and Magerko, B. 2020. What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, 1–16.

MachineLearningForKids. 2017. https://machinelearningforkids.co.uk.

Martin, M.; Chevalier, M.; Burton, S.; Bonvin, G.; Besançon, M.; and Deneux, T. 2023. Effects of Introducing a Learning Robot on the Metacognitive Knowledge of Students Aged 8–11. In *International Conference on Robotics in Education (RiE)*, 169–183. Springer.

Martin, M.; Deneux, T.; and Chevalier, M. 2024. From Automaton to AI Robot: the Added Value for Learning. In Balogh, R.; Obdržálek, D.; and Fislake, M., eds., *Robotics in Education*, 403–410. Cham: Springer Nature Switzerland. ISBN 978-3-031-67059-6.

Ojeda-Ramirez, S.; Rismanchian, S.; and Doroudi, S. 2023. Learning About AI to Learn About Learning: Artificial Intelligence as a Tool for Metacognitive Reflection.

Simons, C.; Metzger, S. R.; and Sonnenschein, S. 2020. Children's metacognitive knowledge of five key learning processes. *Translational Issues in Psychological Science*, 6(1): 32.

TeachableMachine. 2017. https://teachablemachine.withgoogle.com/.

Tensorflow Playground. 2016. Tinker With a Neural Network Right Here in Your Browser. http://playground.tensorflow.org.

Touretzky, D.; Gardner-McCune, C.; Martin, F.; and Seehorn, D. 2019. Envisioning AI for K-12: What should every child know about AI? In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 9795–9799.

Vittascience. 2021. https://en.vittascience.com/ia/.

Walter, W. G. 1950. An imitation of life. *Scientific american*, 182(5): 42–45.

Williams, R.; Park, H. W.; Oh, L.; and Breazeal, C. 2019. Popbots: Designing an artificial intelligence curriculum for early childhood education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 9729–9736.

Wong, G. K.; Ma, X.; Dillenbourg, P.; and Huan, J. 2020. Broadening artificial intelligence education in K-12: Where to start? *ACM Inroads*, 11(1): 20–29.

Zhang, Z.; Willner-Giwerc, S.; Sinapov, J.; Cross, J.; and Rogers, C. 2022. An Interactive Robot Platform for Introducing Reinforcement Learning to K-12 Students. In *International Conference on Robotics in Education (RiE)*, 288–301. Springer.