“Which food do you like?” Tinkering with Neural Networks using AlphAI software and LEGO Spike

Thomas Deneux, Amber Jones and Amy Eguchi

Abstract

Teaching the fundamentals of AI and machine learning to K-12 students presents unique challenges, as these concepts are often too abstract, especially for younger students. While various tools have been developed, explaining how neural networks work in age-appropriate ways remains difficult. The Neuron Sandbox was developed to address this issue, allowing students to explore a singular neuron contextualized through relatable tasks such as “making a peanut butter and jelly sandwich.” Building on this inspiration, the paper presents a hands-on activity using the AlphAI educational software we developed, to make neural networks more tangible through robotics. Students explore the neural network's processing of LEGO sensor inputs. They can either manually edit the network’s weights and thresholds or train them through supervised learning. Live visualizations in 1D and 2D graphs help them interpret how input influences output. The activity provides a guided progression from simple one-input decisions to complex functions requiring hidden neurons, making neural networks both accessible and meaningful for learners. This work was a collaboration between educators from the AI4GA program and researchers. The educators’ feedback guided iterative improvements to AlphAI’s interface and its curriculum, making the platform more engaging and developmentally appropriate for teaching foundational AI concepts.

Introduction

AI education - not only teaching how to use AI, but how AI works and how to build AI - is a major challenge to society’s harnessing of this technology (Touretzky et al. 2019). Since modern AIs rely on Machine Learning, it is important then to teach how an *AI model learns from data*. Several web interfaces have been designed to teach these core concepts to K-12 students. The Neuron Sandbox facilitates understanding by allowing young students the opportunity to play with the most iconic *AI model*, neural networks. Students engage the gamified neural network by setting the weights and activation threshold of a very simple network with 2 input neurons and 1 output neuron (“Neuron Sandbox” 2024; D. S. Touretzky, Chen, and Pawar 2024; D. Touretzky et al. 2025). Several platforms sharing the same principles let students focus on how the model *learns from data*, by letting them train their own model for recognition of images, sounds, and other data types (Lane 2017; TeachableMachine 2017; Cognimates 2018; Vittascience 2021; Carney et al. 2020); but the model itself, more complex, is kept as a black box. Tensorflow Playground (Smilkov and Carter 2016) features a more complex neural network, still with 2 inputs but hidden layers of neurons can be added. To illuminate how it learns from data, it features abstract visualizations of how the neural network succeeds (or not) to generalize label assignments from the training data to the full 2D space. We developed the AlphAI software (Absalon and Deneux n.d.; Martin et al. 2023; Kong and Yang 2023), which provides similar interfaces with increased versatility, and natively integrates several robotic platforms that the neural network can drive, taking the robot’s sensor data as input and using its output to select its actions.

Indeed, educational robotics tools, with their foundation in constructionism, are known to support students’ acquisition of abstract concepts. Constructivist theory suggests that knowledge is developed by actively being constructed and reconstructed through one's experience and direct interaction with the world (Ackermann 1996). Educational robotics is a learning tool that provides hands-on opportunities supporting students’ learning of abstract concepts in tangible and interactive ways (Eguchi 2012, 2024; Karalekas, Vologiannidis, and Kalomiros 2023). Thus, integrating educational robotics tools with AI-enhanced tools can support student learning with hands-on exploratory approaches to ensure their tangible knowledge construction through their interactions with the robotics tools.

Here, we present a new activity of 2 to 4 hours using the AlphAI software together with LEGO Spike robotics set, which was co-designed with educators from AI4GA. This activity is strongly inspired by the Neuron Sandbox interface and activities used in the AI4GA program. As in Neuron Sandbox activities, students will manually edit neural networks to solve challenges and in the most advanced part of the activity, they will analyze 2D graphs of the network input-ouput transfer function. AlphAI, however, allows students to engage the neural network in a “live” reacting robot. Instead of manual editing it will be possible to use supervised learning to train the network using user-collected data, and more complex networks will be studied, including with a hidden layer.

Resources and Activities

A food-detecting LEGO Spike robot

Students build a very simple robot (see Figure 1), using the Spike Prime hub. A LEGO figure sits on top of the hub, and two sensors will detect some "food" presented to the figure: a touch sensor detects a "pizza" LEGO piece and a color sensor detects a "vegetable" LEGO piece. The neural network has one or two input neurons fed by these sensors in an on-off manner (0 for absence, 1 for presence) and one output that turns on a happy face on the hub. It is also possible to adapt the activity for the Spike Essential kit alone; the Spike Essential hub does not have an integrated LED display, and the LED matrix included in the set cannot be used because the 2 ports of the hub will already be used by the 2 sensors; instead, it will be possible to use the status LED of the hub, turn it green to replace the happy face (unfortunately this won't be as demonstrative), and off, or red, when the output is off. For each scenario, we used a different figure, but the Spike Prime set has only two LEGO figures; also, the LEGO piece used to represent a vegetable belongs to the Spike Essential kit. Thus, if one needs to stick to the Spike Prime kit, the vegetable piece can be replaced by a small yellow bar which will be called a "banana", or a red one which will be called a "pepper", and the same figure shall be used to represent different tastes (for example on different days/moods of the same person).

Session 1: Manual edition of the Neural Network

1a Discovery of the Neural Network - logic gate "pizza". When starting the activity, users load a first pre-set configuration, and the software interface shows a very simple neural network with one input neuron and one output neuron (Figure 1A). The input neuron will reflect the state of the LEGO Spike touch sensor, playing here the role of a pizza detector in a pizza rack: if no pizza is detected, the input neuron's value is 0; if a pizza is detected, the value is 1 (Figure 1B). The output neuron will trigger the state of the Spike hub LED display: if activated, the hub will display a happy face. Otherwise, the hub will not display anything.

A first LEGO figure, called "Joshua," is placed atop the hub: Joshua is hungry and would like to eat pizza (Figure 1C). Students are told to set the weight of the connection between the input and output neuron (initially equal to zero) to a value such that the hub will reflect Joshua's mindset. The solution is shown in Figure 1C: by setting a connection weight of 1 (more generally, any weight value > 0.5 will work), we have the hub smiling when there is a pizza in the rack (top), and not smiling otherwise (bottom).

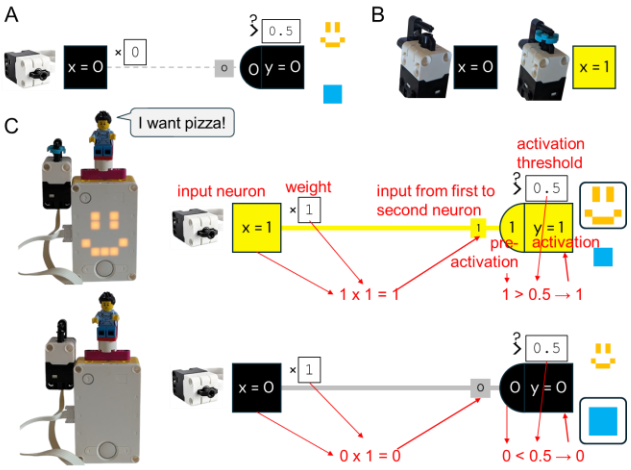

Figure 1: Very first scenario "Joshua wants pizza" with only 1 sensor. (NB: neural network displays from the software interface have been edited to fit space constraints inside figures; red marks do not appear on the software interface).

The red marks alongside the neural network are not shown in the software interface; rather, they show how the teacher should comment on what is happening "inside" the neural network. First, the value of the input neuron (0 or 1) is multiplied by the weight (1), yielding the output neuron's pre-activation value (0 or 1). Second, the output neuron gets activated (and the hub smiles) if and only if this pre-activation value is higher than the output neuron's threshold (0.5). Here, the pre-activation and activation of the output neuron are the same, but this won't always be the case thereafter.

1b Introducing the 2x2 table of logic gates - logic gate "vegetable". In the second scenario, "Isabel wants vegetables," students will use a second sensor, the Spike color sensor, to detect the LEGO pieces representing vegetables (Figure 2A). Now, there are two input neurons: the "pizza" neuron x_1 from the previous scenario, and the new "vegetable" neuron x_2. Students are tasked to edit the network weights to make the hub smile only if a vegetable is present, regardless of the status of pizza: they do it by setting a connection between input x_2 and output y, but keeping zero connection between x_1 and y (Figure 2B). At this stage, the teacher makes them realize that the neural network achieves a function over its two inputs. Since each input is 0 or 1, the function is represented by a 2x2 table (Figure 2C; the table cell with thick borders represents the input configuration shown in Figure 2B).

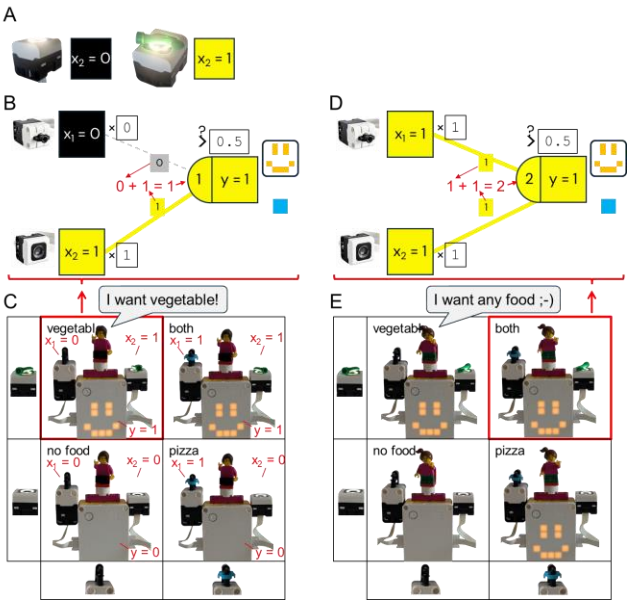

Figure 2: Scenarios “Isabel wants vegetables” and “Jude wants any food” (OR gate).

1c Logic gate “pizza OR vegetable”. With the knowledge they accumulated, students can now set the network’s weights to implement (in Machine Learning terminology, to “model”) the logic gate OR, during the third scenario, “Jude wants any food” (i.e., “pizza OR vegetable”). They need to set positive connections (at least 0.5) between both input neurons and output y (Figure 2D and E). At this point, the teacher guides students to explain the logic of the chosen weights in both the “Isabel wants vegetables” and the “Jude wants any food” scenario. This ensures that there is understanding of the values of the correct weights, even if students relied on guessing and checking to solve.

1d Activation threshold - logic gate “pizza AND vegetable”. After editing and understanding the weights, students need to understand a threshold to solve the next scenario, “Larry is very hungry (and wants both pizza and vegetables)”. The smile should not be activated if only one input is on, so if students connect inputs to output with weight values of 1, they need to change the threshold value from its default value of 0.5 to a value above 1 (for example, 1.5 as in Figure 3). Another solution, without changing the default threshold value of 0.5, would be to set connection weights below 0.5 (for example, 0.4). Whichever solution is found, it necessitates a good understanding of the output neuron’s activation mechanism.

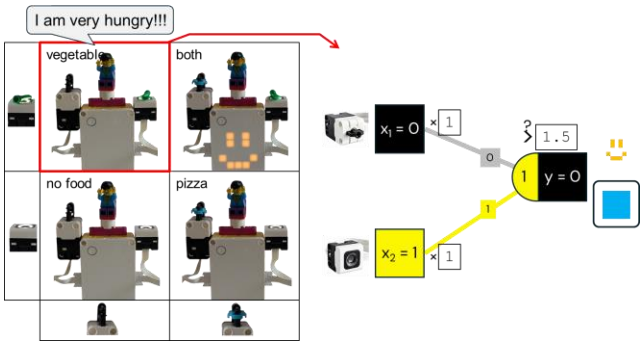

Figure 3: Scenario “Larry is very hungry” (AND gate)

Session 2: Discovery of supervised learning and of state space graphs

2a Supervised learning. So far, students had to manually set connection weights and activation threshold in order for the neural network to model the desired logic function (i.e., the logic gate defined by the tables, such as in Figures 2, 3). However, the power of neural networks resides in the fact that the data scientist *does not need* to find the appropriate parameters. Rather, parameters are *learnt* from data using Supervised Learning.

Therefore, students are now invited to revisit the previous scenarios, and this time, let the supervised learning find the appropriate weights. They switch from *manual edition* to *learning* mode in the AlphAI main control panel (Figure 4A), and set the main display mode to show “Training data and N eural network”. This leads the software to show an additional “training data” display (top part in Figure 4B and C) that is similar to those of Teachable Machine and other interfaces. Now, students can acquire data by pressing the “happy face” or “no display” icons at the right side of either the training data or the neural network. Each time they do so, a new data point is added to the training data, consisting of the value of the sensors at the time of the icon press, and the identity of the pressed icon. This data point appears on screen as a new entry inside the training data section of the selected action (orange frame for the happy face, blue for no display). Immediately, weights and activation threshold of the neural network are automatically modified so as to model a transfer function that matches the examples provided in the training data. Students will be invited to repeat the 4 previous scenarios in learning mode. In Figure 4B, the AI was trained for Jude’s OR gate (“I want any food”), and in Figure 4C, for Larry’s AND gate (“I am very hungry”). In the training data display we see how the training differed for the cases where only one food is provided and in the neural network display we see how the connection weights were estimated to similar values, whereas the main difference resides in the activation threshold being set much higher in the case of the AND gate, similar to what was experienced during the manual edition.

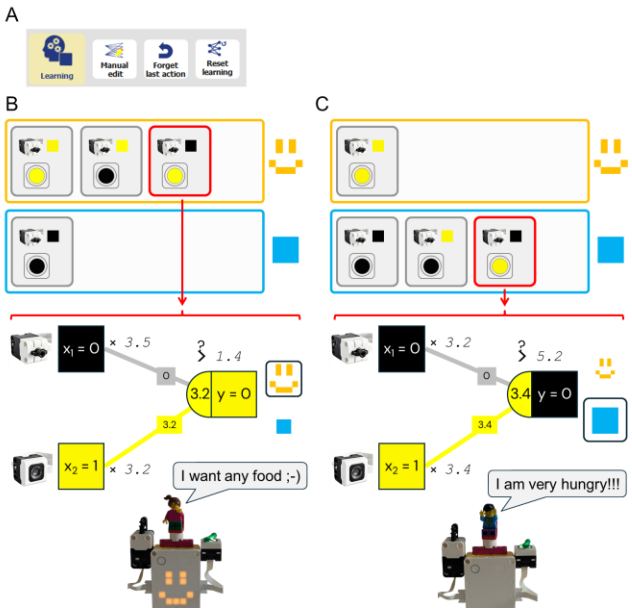

Figure 4: Using supervised learning to automatically train the neural network (revisiting OR gate and AND gate).

2.2 2D graphs. For the sake of keeping the activity simple, we used a robotic configuration where the two sensors are binary: inputs x_1 and x_2 can only be 0 or 1 depending on the detection of pizza and vegetables. However, to fully understand the action of the neural network, it is pertinent to study how its output varies for any real values of $x_1$and x_2. The state graph displays as an image the output y as function of x_1 and x_2, using two different colors for the two possible values 0 and 1. It includes the output of the "logic gate" at positions (0,0), (0,1), (1,0) and (1,1) (see Figure 5 top-left). In AlphAI software, the state graph when shown is positioned above the neural network, such that as soon as network parameters are updated, the graph updates. Students are invited to repeat the 4 previous scenarios and observe how this moves the separation between the blue (no activation) and orange (activation) regions. They will observe that if x_2 makes no connection to output (scenario "Joshua wants pizza"), output happiness depends only on x_1 and the separation is vertical. Similarly the separation is horizontal in "Isabel wants vegetable" case, and becomes diagonal in Jude's OR gate and Larry's AND gate cases. And when transforming and OR gate into an AND gate by increasing the activation threshold, the separation moves upper-right. They will also observe how, when using supervised learning to train the network, the separation lines move during the course of training, until reaching positions close to those obtained when manually editing.

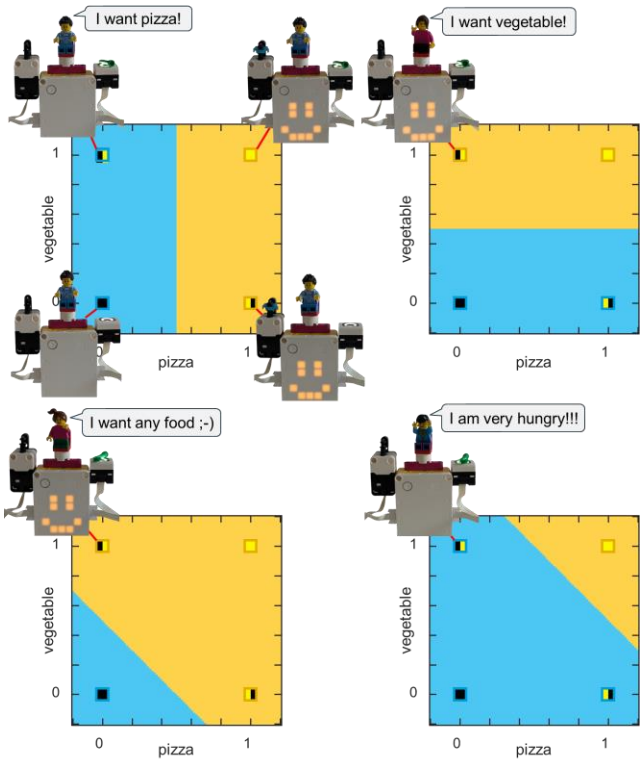

Figure 5: Visualizing the neural network generalization in the 2D state graph

Session 3: Understanding non-linearity and the need for hidden neurons - logic gate "pizza XOR vegetable"

3.1 Discovering negative weights - Logic gates "pizza AND NOT vegetable" and "NOT pizza AND NOT vegetable"

Now, students will benefit from the help of the state graph to solve the next scenarios' challenges. They are asked to always start by trying to set the appropriate connections and threshold manually, and only after they find a solution, do they check what kind of solution supervised learning finds. The first one is "Reggie can't stand veggies." (Only accepted food is pizza without vegetables, Figure 6A): this requires the pizza neuron to form a positive connection to output, but the vegetable neuron must form a negative connection, such that when both are activated, the vegetable cancels the effect of the pizza! (see bottom of Figure 6A). And the second is "Kevin is fasting": now both foods should make negative connections since they both remove Kevin's happiness from fasting. However, what is more difficult to find out might be how to get this happiness activated when there is no food. Teachers guide students to consider that when there is no food, the pre-activation value is zero, so in order for this pre-activation to be greater than the activation threshold, the activation threshold must be negative!

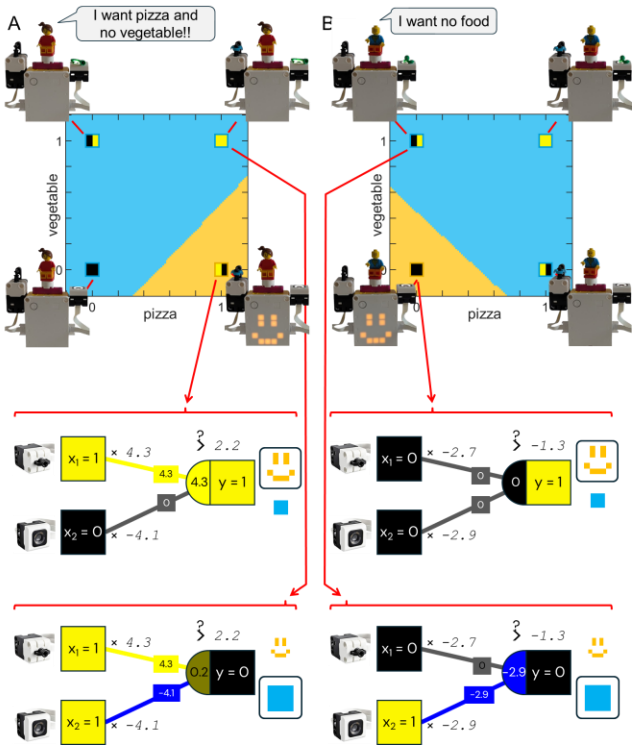

Figure 6: Scenarios "Reggie can't stand veggies," and "Kevin is fasting" (using automatic learning mode).

3.2 The limits of linear neural network - Logic gate XOR

The next and last scenario "Alice does not mix" is a large step of difficulty above: Alice will be happy with either vegetables or pizza, but doesn't want both together. For a predetermined time, teachers should allow students to explore possible weights, thresholds, and experiment with the inputs here, using their knowledge from the previous scenarios to inform their attempts both using manual editing and supervised learning: However, they will never find a solution with the previous methods because it is not possible. Here the teacher can challenge students to express why the neural network fails. Indeed, from what has been learned so far, students can reason that if the pizza neuron connects positively to the output neuron, the presence of pizza can only increase the output activation. Conversely, if it connects negatively, the presence of pizza can only decrease the output activation. So, there exists no configuration which allows pizza to have a "positive" effect on the outcome (switch from unhappy to happy) when there is no vegetable, and at the same time have a "negative" effect (switch from happy to unhappy) when there is already a vegetable present. In more technical terms, the pre-activation value is a linear function of the inputs, and the activation value is a monotonous increasing function of the inputs, so any function which is not monotonous (going "sometimes up, sometimes down") cannot be modelled with such a neural network.

3.3 Solving by adding a hidden layer

It will be possible however to model Alice's XOR gate by adding a layer of intermediary neurons that will break these linear and monotonous behaviours, by letting inputs form both positive and negative "paths" to the output. Students are told how to add a layer of two neurons in the AI configuration tab and asked to make the appropriate connections to have the first of them implement "pizza and not vegetable", the second "vegetable and not pizza", and the output neuron to make an OR between the two intermediary neurons. This results in the state graph and neural network as in Figure 7A and C, implementing the desired XOR behavior!

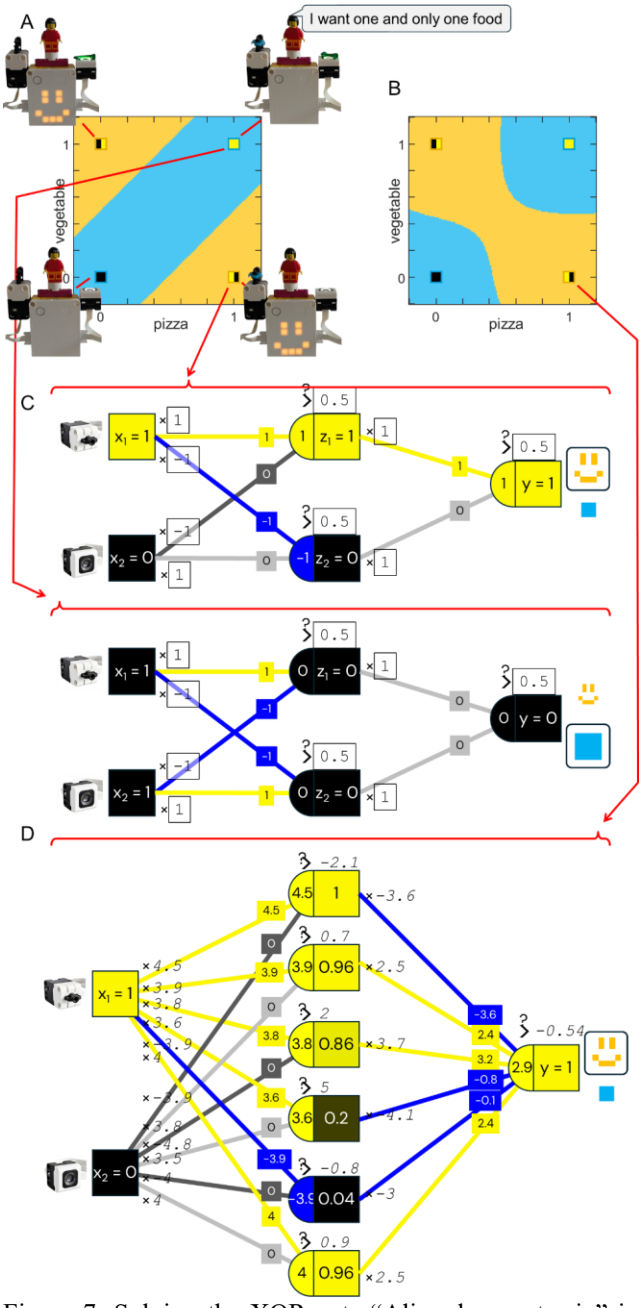

Figure 7: Solving the XOR gate "Alice does not mix" in manual edit (A,C) and in automatic learning modes (B,D).

Now if students try to repeat the network setting in learning mode, they will experience failure. This is because (1) the threshold activation mechanism does not let the learning procedure "feel" whether it is better to increase or decrease the connections on the left of the threshold (in technical words, it does not let the gradients retro-propagate), and (2) to give more chances for both positive and negative "paths" between input and output to form during learning, it is better to have more intermediary neurons, which will each be initialized differently. To fix this, in the AI configuration tab they change (1) the activation method from "Sharp threshold" to "Sigmoid threshold" and (2) the number of hidden neurons from 2 to at least 6. Now learning should succeed (Figure 7B and D), and we get some understanding of what is this "Sigmoid threshold": neuron activations in the hidden layers are not clearcut 0 or 1 (Figure 7D), they are rather "close to 0", or "close to 1", or even can be in the middle between 0 and 1 if pre-activation is close to 0. Consequently, the separation in the state graph is smooth (Figure 7B) contrary to the straight lines in the case of the previous manual editing (Figure 7A).

These final scenarios introduce students to advanced features that they may enjoy exploring further!

Going further

The AlphAI neural network interface and integration of machine learning and robotics allows for fun and interesting extensions to the activities, such as:

• Visualizing the pre-activation function in the 1D graph. When there is only one input, the 2D graph becomes a 1D graph, i.e., there is only one input x, which varies along the x-axis of the graph. In such a case, the AlphAI software puts more information inside the graph, namely, the y-axis is used to display the output neuron's pre-activation (x × weight) as a function of input x. This is a straight line passing through the graph origin (0,0). When this line is above the activation threshold (represented by a horizontal line), the neuron is activated (orange background).

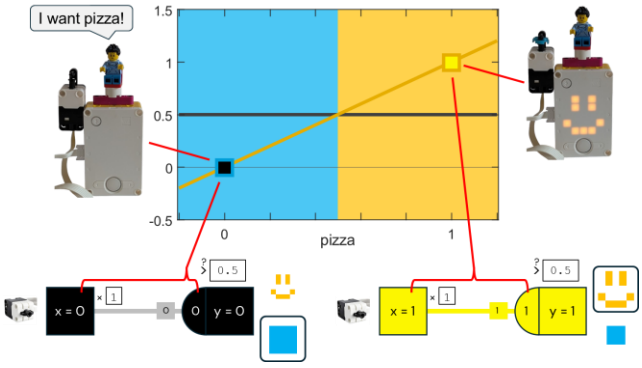

Figure 8: Visualizing the neural network generalization in the 1D state graph

• Use a motor instead of the hub display for more "animated" scenarios. For example, a motor could move the LEGO figure forward when he is willing to eat the presented food, and backward otherwise.

• Use the touch and color sensors in continuous mode, such that the input neurons will have continuous values between 0 and 1 depending on the pressure applied and on the quantity of light reflection. This will make all values in the state graph reachable by actual sensor states. The contextualizing story must be adapted then: for example a different pizza rack should be built that can accommodate more than 1 pizza, and the photosensor should also be positioned in a way that measures the precise quantity of vegetables. Then, new challenges can be defined regarding the ideal menu of different LEGO figures. The more complex the pattern of accepted menus, the more hidden neurons will be needed!

• Teachers might be interested in completely bridging these activities with student experiences in platforms such as Teachable Machine. The main difference is that the activity proposed here uses only 2 single-value sensors, whereas Teachable Machine applies more complex models on rich data such as images and sounds. It is possible in the Sensors configuration tab of the AlphAI to replace the simple sensors with the PC webcam, then configure a more complex neural network in the AI configuration tab (including using the pre-trained Resnet18 model) and train this model to turn the hub happy when the camera sees images of pizzas or vegetables!

Discussion

Thanks to a tangible approach, we make many concepts of AI and Machine Learning accessible at the K12 level:

• The calculations inside a small neural network.

• The concept that the same neural architecture can be tuned to model very different functions (here, "logic gates"). This is the essence of an AI model: a function that can be tuned by settings appropriate parameters values.

• How an AI can drive a physical robot relying on sensors (providing input to AI) and actuators (activated by AI output).

• How to train an AI model using data consisting of example input/output pairs, and how this training consists precisely in setting the weights they had to struggle before setting manually.

• Visualization of the model generalization in a graph (if the number of inputs is one or two, as, of course, visualizing an ND space gets much more difficult when there are more inputs!). It is also very instructive to visualize how this graph evolves during learning (as in the Tensorflow Playground interface).

• Why layers of hidden neurons are necessary, and an understanding of the difference between a linear and a nonlinear function (in a linear setting, the same piece of food

will always have the same effect, either positive or negative, on output happiness; in a nonlinear setting, the same piece of food can have both positive and negative effects depending on the context. Of course, real-world problems are essentially nonlinear and require very large multilayer neural nets!).

• The existence of more complex factors for successful training is revealed, such as choosing the appropriate activation function or learning rate. Without going into more details, this gives them a glimpse of how modeling complex functions requires know-how that can be acquired by studying and experiencing.

These concepts might seem complex for the K-12 level. However grasping them will really help them developing an engineering mindset, giving them confidence they can harness the technology, instead of considering it as an out-of-reach mystical object.

And grasping these concepts is in fact at reachable distance thanks to our efforts to make them very concrete:

• Use of a fun LEGO robot: students manipulate with their hands the neural network sensor inputs and see its output physically.

• Contextualization with the "Which food do you like?" story, inspired by the Neural Sandbox "peanut butter and jelly" and other activities. Assigning a different LEGO figure per food preference also contributes to build the story and make the abstract concepts of "models", or "logic gates" appealing.

• Tinkering of the neural models, going back and forth between manual edition and supervised learning, and seeing the effects of parameter changes on the model generalization as it appears in the state graph (displacement of the separation border, etc.).

• Furthermore, the interaction with AI4GA educators led to thorough improvements in AlphAI's visualization of the neural network. All steps of the calculations now appear on screen, with appropriate colors, font size, and labels. A strong coherence with the Neural Sandbox interface was sought as well, to allow using both tools with the same students.

The interaction with AI4GA educators led to two major improvements in the AlphAI interface that are interesting to discuss here. First, we followed the choice of Neural Sandbox to show the nonlinear neural activation in the form of an activation threshold rather than as a classic nonlinear activation function. This simplifies understanding, and also allows not using neuron biases. It is interesting indeed to note that activating the output neuron when pre-activation is greater than a threshold (for example, 0.5) is equivalent to, but much easier to explain, adding a bias value of -0.5 then applying a Heaviside activation function. The AlphAI interface has been adapted to be capable of displaying either classic neural biases and activation functions, or simpler activation thresholds. Second, it appeared very important that *every* detail of an artificial neuron calculation appears on screen, including: individual inputs received from pre-synaptic neurons, pre-activation value, neural bias or activation threshold parameters, activation value, weight parameters of the connections to post-synaptic neurons. Displaying so much information on screen required designing carefully how to position elements, use appropriate color or gray shades, font types and sizes, boxes or not.

References

Absalon, Marie, and Thomas Deneux. n.d. AlphAI: Teaching AI Algorithms to K12 in a Concrete MannerWith a Graphic Software and Learning Robots. In *Proceedings of the AAAI Symposium on Educational Advances in Artificial Intelligence*, Philadelphia, 2025

Ackermann, E. K. 1996. Perspective-Taking and Object Construction: Two Keys to Learning. In *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World*, edited by Y. Kafai and M. Resnick, 25–37. Lawrence Erlbaum Associates.

Carney, Michelle, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 1–8. New York, NY, USA: ACM.

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

Eguchi, A. 2012. Educational Robotics Theories and Practice: Tips for How to Do It Right. In *Robotics in K-12 Education: A New Technology for Learning*, edited by B. S. Barker, G. Nugent, N. Grandgenett, and V. L. Adamchuk, 1–30.

Eguchi, A. 2024. Revisiting the Pedagogy of Educational Robotics. In *Lecture Notes in Networks and Systems*, 81–92. 747.

Karalekas, Georgios, Stavros Vologiannidis, and John Kalomiros. 2023. Teaching Machine Learning in K–12 Using Robotics. *Education Sciences* 13 (1): 67.

Kong, S. C., and Y. Yang. 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*.

Lane, D. Machine Learning For Kids. 2017. https://machinelearningforkids.co.uk.

Martin, Marie, Morgane Chevalier, Stéphanie Burton, Guillaume Bonvin, Maud Besançon, and Thomas Deneux. 2023. Effects of Introducing a Learning Robot on the Metacognitive Knowledge of Students Aged 8--11. In *International Conference on Robotics in Education (RiE)*, 169–83. Springer.

Chen, A., Pawar, N., and Touretzky, D.S. Neuron Sandbox. 2024. https://www.cs.cmu.edu/~dst/NeuronSandbox/.

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

Smilkov, D. and Carter, S. 2016. *Tensorflow Playground*. https://playground.tensorflow.org/.

Touretzky, David, Christina Gardner-McCune, William Hanna, Angela Chen, and Neel Pawar. 2025. Thinking Like A Neuron in Middle School. In *Proceedings of the AAAI Symposium on*

Educational Advances in Artificial Intelligence, Philadelphia, 2025.

Touretzky, David S., Angela Chen, and Neel Pawar. 2024. Neural Networks in Middle School. *ACM Inroads* 15 (3): 24–28.

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