Neural Networks in Middle School

By David S. Touretzky, Angela Chen, and Neel Pawar, *Carnegie Mellon University*

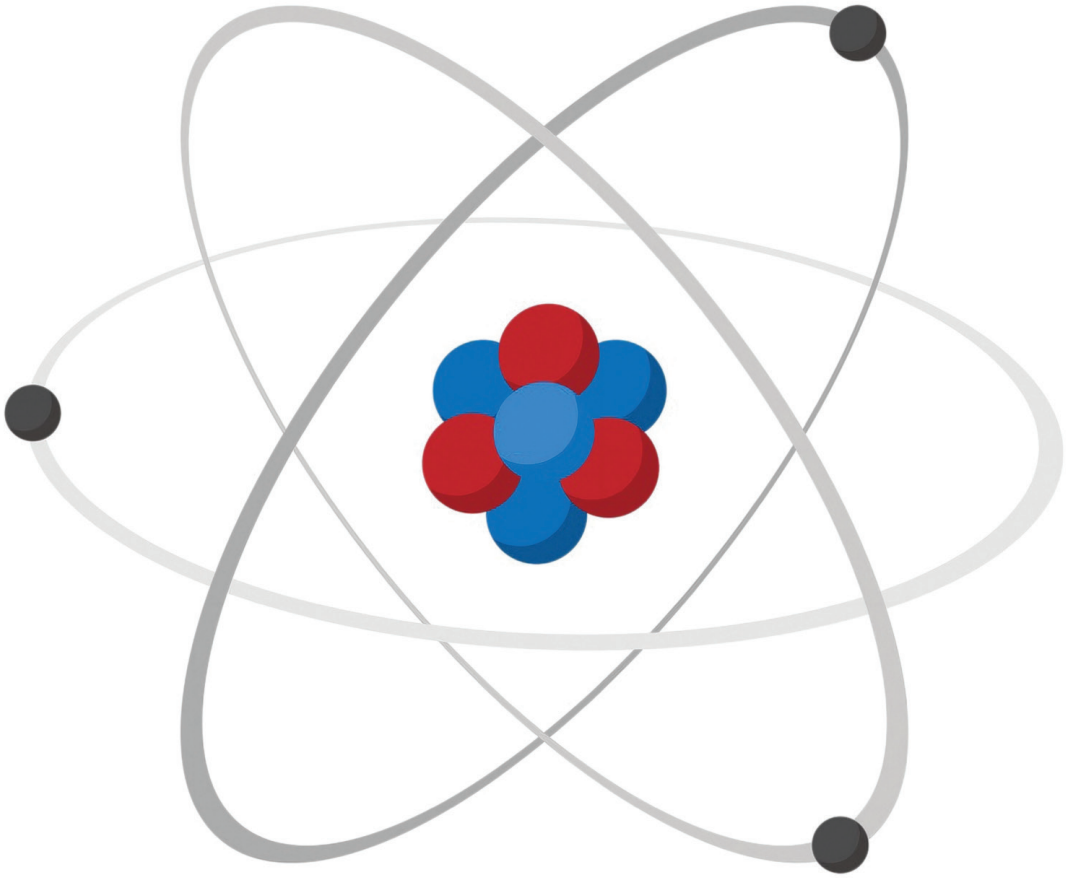

Figure 1

The planetary model of the atom in Figure 1 is alive and well in middle school science class [5]—and in popular iconography—despite most educated adults' awareness of its shortcomings. The model persists because it is easily visualized, intuitively understandable, and expresses important truths.[1] Models don't have to get everything right to be useful. Middle schoolers would be overwhelmed by a more correct description of electron orbitals as probability densities satisfying the Schrödinger equation. Better to just show orbitals as ellipses.

In the current era, when immensely powerful AI technologies built on neural networks are rapidly disrupting the world, K-12 students need age-appropriate models of neural networks just as they need age-appropriate models of atoms. We suggest the linear threshold unit as the best model for introducing middle school students to neural computation, and we present an interactive tool, Neuron Sandbox, that facilitates their learning.

[1] Also known as the Bohr model. See [9] for multiple discussions about the continued use of the Bohr model in schools.

WHAT CAN MIDDLE SCHOOL STUDENTS LEARN ABOUT NEURAL NETS?

Present approaches to teaching children about neural networks are often superficial and ineffective (complicated diagrams with little explanation) or seriously misleading (claims that neural networks somehow "work like the brain"). Unplugged activities can avoid those pitfalls. One strategy used in McOwan and Curzon's "Brain in a Bag" [7] has a group of students physically simulate a small neural circuit, with each student playing the role of a neuron and transmitting a "spike" to a downstream neuron by sliding a tube along a segment of rope representing the connection between them. Together the students compute an XNOR function. This gives them some feel for network computation, but it does not develop any problem-solving skills or general insight into how weights and thresholds determine neuron behavior.

Another approach is to provide a high level, abstract description of what the layers of a neural network are doing and ask students to simulate that abstraction. Lindner and Seegerer [6] attempt to explain object recognition in a neural network by having students simulate multiple layers of feature detectors. In the first layer a student converts a photo to a pair of line drawings, in the next another student looks for specific shapes in the drawings (triangular, rectangular, and round shapes), and in the final layer a third student determines the object label based on which shapes were detected. This activity gives a qualitative feel for the hierarchical nature of object recognition. But because these operations are far more complex than what individual neurons compute, and converting a photo to drawings is not actually part of the object recognition process in neural networks, it's unclear what real understanding participants can develop beyond a vague notion of "feature detection."

While there is some value in giving middle school students a qualitative feel for neural networks, they are capable of more rigorous understanding when given the necessary scaffolding. We suggest that an appropriate entry to neural networks in middle school is to explore in depth the reasoning capabilities of a single simulated neuron. Although most middle school students have not yet been exposed to algebra, we have found they can master the behavior of a single neuron when applied to binary decision problems such as "Can I make a peanut butter and jelly sandwich?" The type of neuron to use for these problems is a linear threshold unit (LTU). LTUs compute an activation value as a linear combination (weighted sum) of their inputs and output a 0 or 1 based on whether the activation exceeds a threshold. LTUs were the building blocks of the earliest neural networks such as the perceptron [8,10]. But since modern neural network learning algorithms require differentiable activation functions, today's neural network architectures such as transformers and convolutional networks use the ReLU (Rectified Linear Unit) or some variant of it instead [4]. We stick with LTUs because they allow middle schoolers to explore an easier to understand set of examples based on Boolean logic. Like the planetary model of the atom, historically important but no longer preferred by practitioners, teaching LTUs in middle school to explain neural networks is a forgivable oversimplification. Heavy scaffolding will nonetheless be required.

Will LTUs be welcomed into middle school curriculum standards? There are already calls to integrate AI topics into K-12 computing courses [13]. LTUs might be an attractive choice due to the strong overlap with established middle school math topics, including (1) algebraic expressions using multiplication and addition, (2) inequalities, (3) working with negative numbers, (4) logical reasoning, (5) truth tables, and (6) digital logic. What LTUs add to the mix is a version of decision making through numerical computation.

NEURON SANDBOX

Neuron Sandbox [3] is our interactive tool for exploring the linear threshold unit's decision-making abilities. For middle school students we focus on implementing Boolean functions by specifying weights and a threshold.[2] As shown in Figure 2, the process is highly scaffolded: the entire truth table is visible all the time, and there is a direct comparison of the neuron's actual output with the desired output for each row of the table. Thus, it is easy to see when a problem has been solved correctly, or where the sticking points lie if the correct weights and/or threshold value have not yet been found. Icons illustrate the meaning of the binary inputs and output, e.g., if "Have Peanut Butter" is true we see a peanut butter jar, while if it is false the jar is crossed out. Additional scaffolding helps students see exactly how the neuron's activation value is computed for a given input pattern. In the figure, the mouse hovering over the last row of the truth table triggered blue highlighting and leader lines that unpack the computation for that row. Also visible in the figure is another slight simplification: rather than introducing Σ as the sequence summation operator, we use it as a symbol designating the sum, i.e., Σ denotes the neuron's activation value.

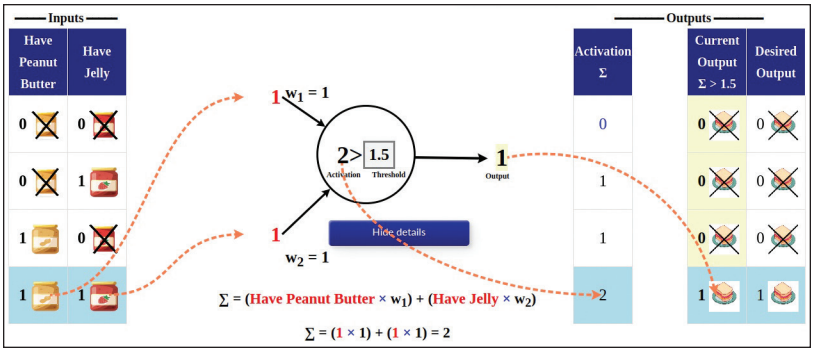

Figure 2

[2] For more advanced students there are other options available, such as real-valued inputs, use of biases instead of thresholds, and visualization of decision boundaries.

Mastering LTUs requires a series of skills. Undergraduates approaching this topic would not have to be taught these skills explicitly because their prior mathematical training has familiarized them with similar types of problem solving. But middle schoolers, having not yet taken algebra, may be encountering these ideas for the first time. Table 1 lists the skills we have identified.

Table 1: Skills required to master linear threshold units.

(1) Mapping an English statement to a logical function
(2) Constructing a truth table
(3) Calculating the activation value of a neuron given its weights and input
(4) Calculating the output value of a neuron given its activation and threshold
(5) Reasoning about how changing a weight or threshold would affect a neuron's behavior
(6) Finding weight and threshold values that allow a neuron to produce the desired output for each possible input

To facilitate learning to map English statements to logical functions, Neuron Sandbox includes a "solve for outputs" mode where students are given a textual description of the function and fill in the truth table themselves by clicking on radio buttons. We scaffold this by generating the rows of the table automatically and providing a "Press to Check" button for students to check their answers. If they get an entry wrong, a "hint" appears to prompt them to rethink their answer. Figure 3 shows how a student could fill in the truth table for AND given the prompt "Can I make a peanut butter and jelly sandwich? I need both peanut butter and jelly," and what they see when they predict the wrong output value for one of the cases.

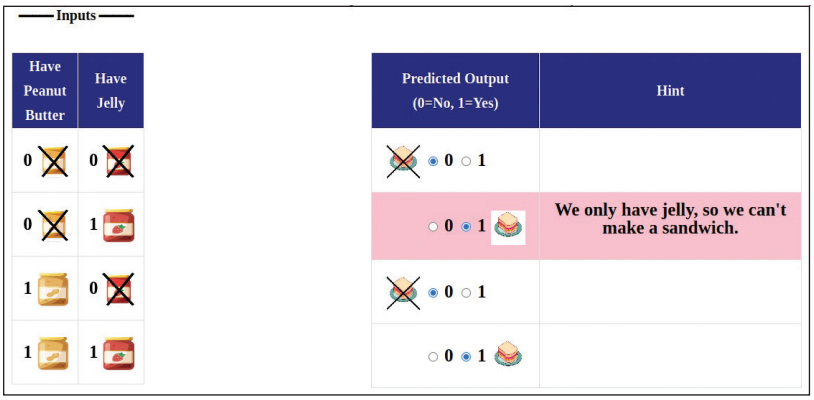

Figure 3

LEARNING PROGRESSION

The easiest functions for students to learn are AND and OR, since they align with common English terms and can be expressed using identical weights of 1. The solution for AND is shown in Figure 2. We define a six-stage learning progression to describe how students can learn to solve AND and OR and then extend their understanding beyond those two functions.

Stage 1: Boolean functions, truth tables, how a linear threshold unit works, and implementation of the AND and OR functions. The weights are 1 and the thresholds are positive: 0.5 for OR, 1.5 for AND. Although we could use integer threshold values (0 for OR, 1 for AND), that solution critically depends on the threshold comparison being a strict inequality, i.e., "greater than" rather than "greater than or equal to." Students who are not mindful of the difference will likely become confused. We have even experienced this ourselves. We therefore believe it is preferable to select thresholds strictly smaller than the minimum required activation value, to forestall this type of misunderstanding.

If the least interested students successfully memorize the weight and threshold values for AND and OR, we might be satisfied. But for more engaged students, it is helpful to explore variants of the standard parameter values. For example, if we doubled the weights, how would the threshold have to change to implement AND and OR correctly? Or if we wanted to use a threshold of 7, what weight values should we use?

If time is limited, completing just stage 1 is a reasonable goal for middle school. While not enough to claim mastery of LTUs, stage 1 is sufficient to give students a feeling for how artificial neurons compute.

Stage 2: Logical negation (the NOT function), a one-input function that demonstrates that weights and thresholds may need to be negative.

Stage 3: Combining negation with AND and OR, yielding NAND (NOT-AND) and NOR (NOT-OR). Students will see that AND and NAND use the same weight and threshold values but with opposite signs (all positive for AND, all negative for NAND). The same symmetry holds for OR and NOR.

Stage 4: Three-input versions of AND, OR, NAND, and NOR demonstrate that the same linear equation for the activation value applies but extended with one more input and one more weight. One more input means the truth table doubles in size, going from 4 rows to 8. With three inputs we can also consider a new function: "at least two," which is true if at least two of its three inputs are true. This would require a complex circuit if implemented using digital logic gates, e.g., $(a \wedge (b \vee c)) \vee (b \wedge c)$, but it is trivial for a linear threshold unit. The "at least two" example points at an important truth about neural computation: it is fundamentally numerical, not digital.

Stage 5: Returning to two-input functions, we now consider asymmetric functions AND-NOT ("x and not y") and OR-NOT ("x or not y"). This introduces non-uniform weights; in previous stages all the inputs had the same weight. Asymmetry means "x and not y" and "y and not x" are distinct functions.

AI technology is changing not only how we teach, but what we teach. Neural networks, machine learning, and generative AI are all finding their way into the middle school curriculum. Neuron Sandbox was created to support this movement. Its design affords learning a simple model of neural computation without even having mastered algebra.

We use the phrasing "y and not x" in place of "not x and y" to avoid the ambiguity between "(not x) and y" and "not (x and y)", something which middle schoolers are likely to trip up on.

Stage 6: Now we consider the limitations of LTUs. The two-input EQUAL and XOR (exclusive-OR) functions cannot be realized using a single LTU. In high school this would be demonstrated geometrically by drawing linear decision boundaries, and for undergraduates we would give a proof by contradiction that no set of weight and threshold values exists that satisfies all four cases. But for middle school we simply tell students that these functions aren't realizable and invite them to try finding weights and a threshold to disprove this. The three-input "exactly two" function (as opposed to "at least two") is also unrealizable, for the same reasons. Students now see that LTUs are not all-powerful and can appreciate why typical neural networks require many neurons. Advanced students might be shown the three-neuron solutions to EQUAL and XOR.

HEURISTICS FOR MASTERY

Expert-level knowledge of a problem domain includes heuristics that speed the search for a solution. In the narrow domain of two-input Boolean functions implemented using an LTU, students can easily be taught these heuristics or discover them for themselves. There are 16 cases, shown in Table 2.

Numerical values: although an LTU's weights and thresholds can be arbitrary numbers, the two-input Boolean functions (except EQUAL and XOR) can all be realized with just a few values:

- Weights are ± 1, or 0 if the input is to be ignored.
- Thresholds are either ± 0.5 or ± 1.5.

Note that this small set of weight and threshold values won't suffice for functions of more than two inputs. For example: "John can have dessert if either he has eaten all his vegetables, finished his milk, and cleared the table, or if the dessert is something healthy." This is an asymmetric four-input Boolean function that can be realized with weights of 1, 1, 1, and 3, and a threshold of 2.5.

Threshold sign: a heuristic for choosing the threshold is:

- If the desired output is 0 when all inputs are 0, the threshold must be positive (or zero). Otherwise, the threshold must be negative.

Constraints on weights:

- If the function is symmetric, the weights should be identical.
- Comparing rows in the truth table, if flipping a single input from 0 to 1 makes the output go from 0 to 1, the weight on that input must be positive. If flipping an input from 0 to 1 makes the output go from 1 to 0, the weight on that input must be negative.

We believe teaching these heuristics explicitly can foster development of the reasoning skills enumerated in Table 1.

WHAT ELSE SHOULD MIDDLE SCHOOL STUDENTS KNOW?

There are three other topics middle school students should learn about when studying neural networks. First, they should understand that more complex computations than the ones they have studied in Neuron Sandbox require collections of neurons wired together to form a network. They should become familiar with the concept of feedforward networks with input, hidden, and output layers.

Second, they should understand that in these large networks, rather than setting the weights by hand, they are set by a learning algorithm. Although the backpropagation learning algorithm is too advanced for middle school, the perceptron learning algorithm for training a linear threshold unit is simple enough that it can be hand-simulated in class. In the AI4GA ("AI for Georgia") project's AI curriculum for middle school

Table 2: All 16 two-input Boolean functions. Shaded functions cannot be implemented by a single LTU.

Zero	a	b	AND	OR	a AND NOT b	a OR NOT b	**EQUALS**
One	NOT a	NOT b	NAND	NOR	b OR NOT a	b AND NOT a	**XOR**

students, Touretzky and colleagues modeled this in an exercise called "Can I pet this?" [1]. Students see how an LTU can learn to classify animals as pettable or not based on weighted inputs from a collection of feature detectors.

Finally, students should understand the distinction between training and application phases in machine learning. Training a neural network requires a learning algorithm and a collection of labeled examples. Once trained, learning ceases and the network is applied to novel inputs to predict their labels. A good way to illustrate this is with Teachable Machine [2], a browser-based machine learning tool for training visual or audio classifiers. Students define a set of classes and collect training data for each class using their computer's camera or microphone. After training, they can test their classifier's accuracy on new images or sounds. Teachable Machine is widely used in K-12 AI education, at many grade levels. Sanusi et al. describe a successful use of the tool with eighth graders in Nigeria [11].

Although the neural network inside Teachable Machine is not visible to users, students can understand that it is a feedforward network with many hidden layers (hence a "deep" network) that were trained by a learning algorithm. The neurons in this network are not precisely LTUs, and the learning algorithm is not exactly the perceptron learning algorithm, but like the elliptical orbitals of middle school atoms, these age-appropriate simplifications will facilitate deeper understanding in later years. High school juniors and seniors taking an AI elective can experiment with true backpropagation learning using an interactive tool such as TensorFlow Playground [12].

CONCLUSION

AI technology is changing not only how we teach, but what we teach. Neural networks, machine learning, and generative AI are all finding their way into the middle school curriculum. Neuron Sandbox was created to support this movement. Its design affords learning a simple model of neural computation without even having mastered algebra. Preliminary experiments with middle school students using Neuron Sandbox suggest this approach can be effective if students are shown step-by-step how to calculate the neuron's activation and how to reason about changes to the weights or threshold affecting the output. We are presently collecting data that will allow us to further refine the design of Neuron Sandbox and determine what additional types of support the software and teachers might provide to students. ❖

Acknowledgments

We thank Christina Gardner-McCune, Will Hanna, and Janet Kolodner for suggestions contributing to the development of Neuron Sandbox and its accompanying curriculum. This research was supported by National Science Foundation awards DRL-2049029 and IIS-2112633.

References

1. "AI for Georgia" Project. https://ai4ga.org; accessed 2024-3-10.
2. Carney, M., Webster, B., Alvarado, I., Phillips, K., Howell, N., Griffith, J., Jongejan, J., Pitaru, A., and Chen, A. "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 (CHI EA '20). (Honolulu, HI: Association for Computing Machinery, 2020): 1–8.
3. Chen, A., Pawar, N., and Touretzky, D. "Neuron Sandbox". https://www.cs.cmu.edu/-dst/NeuronSandbox; accessed 2024-7-17.
4. Goodfellow, I., Bengio, Y., and Courville, A. Deep Learning. (Cambridge, MA: The MIT Press, 2016).
5. Hewitt, P. G. "The Bohr model of the atom." *The Science Teacher* 88, 3 (2021): 14–16.
6. Lindner, A., and Seegerer, S. "AI Unplugged: Unplugging Artificial Intelligence." https://www.aiunplugged.org/english.pdf, (8-11); accessed 2024-6-2.
7. McOwan, P., and Curzon, P. "The Brain-in-a-Bag Activity," https://teachinglondoncomputing.org/resources/inspiring-unplugged-classroom-activities/the-brain-in-a-bag-activity; accessed 2024-6-2.
8. Minsky, M., and Papert, S. Perceptrons: An Introduction to Computational Geometry. (Cambridge, MA: The MIT Press, 1969).
9. Quora. "Why is Bohr's model of the atom still taught in schools when Schrödinger's model is more accurate?" https://www.quora.com/Why-is-Bohrs-model-of-the-atom-still-taught-in-schools-when-Schrodingers-model-is-more-accurate; accessed 2024-3-10.
10. Rosenblatt, F. Principles of Neurodynamics. (Washington, DC: Spartan Books, 1982).
11. Sanusi, I. T., Omidiora, J. O., Oyelere, S. O., Vartiainen, H., Suhonen, J., and Tukiainen, N. "Preparing middle schoolers for a machine learning-enabled future through design-oriented pedagogy." IEEE Access 11 (2023): 39776-39791; https://doi.org/10.1109/ACCESS.2023.3269025
12. Sato, K. "Understanding neural networks with TensorFlow Playground." Google Cloud Blog, July 26, 2016; https://cloud.google.com/blog/products/ai-machine-learning/understanding-neural-networks-with-tensorflow-playground; accessed 2024-7-19.
13. Yongpradit, P. "What is the future of K-12 CS education in an age of AI?" Code.org blog, December 7, 2023; https://codeorg.medium.com/what-is-the-future-of-k-12-cs-education-in-an-age-of-ai-17a03187c005; accessed 2024-7-19.

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dst@cs.cmu.edu

Angela Chen
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
angelac3@andrew.cmu.edu

Neel Pawar
Carnegie Mellon University
Pittsburgh, PA 15213
neelpawar17@gmail.com

DOI: 10.1145/3677610