Beyond Black-Boxes: Teaching Complex Machine Learning Ideas through Scaffolded Interactive Activities

Brian Broll[1] **and Shuchi Grover**[2]

[1]Vanderbilt University
[2]Looking Glass Ventures
brian.broll@vanderbilt.edu, shuchig@cs.stanford.edu

Abstract

Existing approaches to teaching artificial intelligence and machine learning (ML) often focus on the use of pre-trained models or fine-tuning an existing black-box architecture. We believe ML techniques and core ML topics, such as optimization and adversarial examples, can be designed for high school age students given appropriate support. Our curricular approach focuses on teaching ML ideas by enabling students to develop deep intuition about these complex concepts by first making them accessible to novices through interactive tools, pre-programmed games, and carefully designed programming activities. Then, students are able to engage with the concepts via meaningful, hands-on experiences that span the entire ML process from data collection to model optimization and inspection. This paper describes our *AI & Cybersecurity for Teens* suite of curricular activities aimed at high school students and teachers.

Introduction

By 2024, more than eight billion AI-powered digital voice assistants (a number roughly equal to the world's population) will be in use globally (Thormundsson 2022).

There is growing recognition of the need to teach about artificial intelligence and machine learning (AI/ML) at the school level in light of the meteoric growth in the range and diversity of application of machine learning (ML) in all industries and everyday consumer products (Royal Society (Great Britain) 2017; Touretzky et al. 2019b). Efforts to bring AI, especially ML, are being propelled by efforts such as AI4K12 (Touretzky et al. 2019a) and technological developments that allow sophisticated ML models to be run in a web browser making these tools readily available to learners of all ages. While these efforts span a variety of learning goals captured by the AI4K12 "big ideas" framework (Touretzky et al. 2019a), many focus on ML and introduce the ideas surrounding ML, and give students a sense of ML modeling techniques such as classifiers and neural networks through the use of examples and extant pre-trained models and tools such as Google's *Quick, Draw!*, *Teachable Machine*, and *Tensorflow Playground*. The goal in our project is to take these learning experiences a step further and help secondary students build deeper understanding of how AI/ML techniques work and how the machine actually ***"learns"***. Our innovation lies in resolving the challenge of making concepts that involve complex mathematics accessible to students that have not yet learnt those mathematical ideas. Through making real-world connections along with the interplay of mathematics apparent in learner-friendly ways, we contend that students will build deeper and better intuitions of ML techniques. We believe that such deeper understanding also aids a more meaningful interrogation of critical issues such as ethics and bias in AI/ML. We draw inspiration from Bruner, "any subject can be taught effectively in some intellectually honest form to any child at any stage of development." (Bruner 1960) and past work in turtle geometry by pioneers such as Abelson and diSessa that made sophisticated ideas in mathematics and physics accessible to young learners through leveraging multiple representations and programming (Abelson and DiSessa 1986).

In this paper, we describe our *AI & Cybersecurity for Teens* (ACT) sequence of curricular activities for high school classrooms that introduce key AI/ML ideas and techniques such as rule-based AI, classification, decision trees, optimization, gradient descent, neural networks, and adversarial examples through a range of programming activities in NetsBlox (Broll et al. 2017, 2021)—an extension of the block-based programming environment Snap*!* that includes features for easy distributed computing—that help build intuitions and understanding through increasingly deeper levels of engagement with ML algorithms and code. Although students engage in playful explorations through designed non-programming and digital games as well as existing interactive tools as in existing efforts, our activities use these as a gateway to actually lifting the hood on how such models are coded. Our exploratory work involved getting feedback from high school teachers who were trained on these activities through two workshops. Teacher feedback has helped refine these activities. Our next steps involve getting feedback from classroom use by our participating teachers. In our presentation, we hope to provide live, hands-on demonstrations of our designed games and programming projects.

Background & Related Works

Several efforts are underway to build curricula that enable students to interact with AI (Druga, Otero, and Ko 2022).

There are curricula being designed for school-going students of all ages. Example efforts targeting diverse goals include broad AI literacy of AI/ML techniques such as decision trees, supervised learning, neural networks, and GANs along with related issues of ethics, awareness, and careers (Lee et al. 2021), ethics-focused experiences (Payne 2019) for middle school students, engaging high school girls in socially-relevant AI experiences (Alvarez et al. 2022), teaching AI basics from a computer science topics lens (Burgsteiner, Kandlhofer, and Steinbauer 2016), connecting AI tools and examples to core subjects (Van Brummelen and Lin 2020), bringing AI education to elementary science classrooms (Glazewski et al. 2022), and AI literacy among young children through interactions with and lo-fi prototyping of AI agents (Druga et al. 2019; Druga and Ko 2021).

However, research on appropriate strategies and pedagogies for teaching ML, especially to K-12 students is still evolving with little clarity on how—and at what depth—to teach these complex ideas to younger learners (Evangelista, Blesio, and Benatti 2018; Hitron et al. 2019). Studies have demonstrated that children as young as 10-13 years of age are able to understand key ideas of ML classification (specifically data labeling and evaluation) through appropriately designed experiences with pre-trained models (Hitron et al. 2019). Hitron and colleagues (Hitron et al. 2019) also assert that merely interacting with AI agents and ML models does not expose students to the underlying processes of ML, and argue for helping students build appropriate mental models of ML through "uncovering the black-boxed processes". Although programming-focused curricular activities that leverage block-based programming environments (Kahn and Winters 2017, 2021; Kahn et al. 2018; Lane 2021) notably take steps toward uncovering ML black-boxes, most rely on pre-trained models and APIs to demonstrate ML. A few notable exceptions include (Jatzlau et al. 2019; Guerreiro-Santalla et al. 2022) which present AI techniques such as Q-learning and clustering in a hands-on, project-based curriculum. Our work follows a similar approach but attempts to explore deeper into fundamental concepts like optimization then leverage this insight to inform rich discussions around cybersecurity, ethics, and bias.

ACT Curricular Activities

Leveraging Block-Based Programming in NetsBlox

NetsBlox is a block-based programming environment based (Broll et al. 2017) on Snap*!* which was designed to make networking & distributed computing concepts accessible to novices. It has also been used to introduce other advanced CS topics, such as the Internet of Things and robotics, to young learners (Broll et al. 2021). The main technical extensions of Snap*!* (Harvey and Mönig 2017) are the networking primitives for *message passing* and *Remote Procedure Calls* (RPCs). Message passing allows users to send messages to remote computers via the internet. This enables students to make engaging multi-user applications.

RPCs allow users to invoke functionality on the NetsBlox server. Conceptually, they are similar to custom blocks except the code is (usually) implemented in another language and runs on a remote server. RPCs with similar functionality are grouped together into NetsBlox *Services*, such as Google Maps, climate data, and the `ParallelDots` API for sentiment analysis (used in the ACT cyberbullying text classification activity). Documentation for RPCs is integrated into NetsBlox and contains helpful information with examples.

As an introduction to programming in NetsBlox, ACT starts with a chat application, a project that is extended in ACT activities. The chat application is a simple message passing application with a single server and potentially many clients. The instructor and students create their own projects and the students are first tasked to send a message to the teacher's project. After sending a message to the teacher, they extend their projects so they can view the messages of their peers from their own project. This involves adding a registration step with the server so the server can then relay chat messages to the list of registered client projects.

An example chat client is shown in Figure 1. This client connects to a hard-coded server address, "ACTChatServer@brian", and then repeatedly sends user input to the server. When it receives a chat message, it simply displays the message on the screen. Not only is this project simple (requiring fewer than 20 blocks for the client) but it also provides a foundation from which other cybersecurity topics can be explored including Denial of Service attacks and identity spoofing. This project also can be easily connected to cryptography and/or cybersafety by encrypting messages before sending them or impersonating another user by changing the "sender" field.

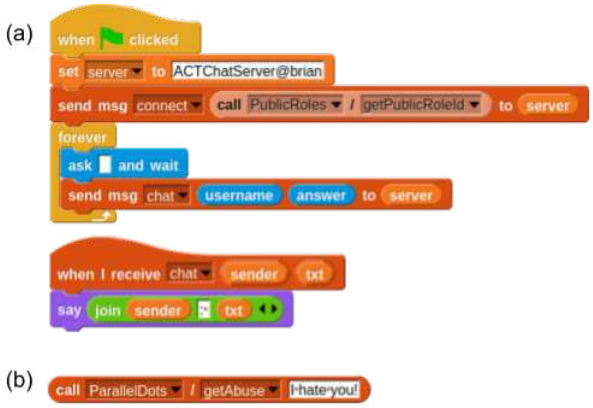

Figure 1. a) Completed client code for a chat app; b) An example RPC call (to ParallelDots, to check if text is abusive).

Design Approach

ACT activities evolved following feedback from 7 high school teachers who participated in a preliminary PD workshop, and then from 5 high school teachers who aim to integrate these activities in their classroom teaching. The following pedagogical strategies guide our work–

Learning in Context. The learning sciences strongly advocate for learning to occur in context (Bransford et al. 2000; Brown, Collins, and Duguid 1989) and through making real-world connections to the content (Gainsburg 2008).

For this reason, and because cybersecurity is another growing subfield of computing that we believe students must get exposure to, we situate our ML programming projects and activities in the context of real world, topical cybersecurity issues such as detecting and mitigating cyber fraud and Denial of Service (DoS) attacks, classifying phishing emails, Twitter and registration bots, and understanding how deep fakes are created (as examples of generative adversarial models).

The design of relevant AI activities to be embedded within cybersecurity learning are guided by a set of "big ideas" or learning goals for our innovative designs. Through ACT activities, students will develop a sense for:

1. How AI/ML plays a role in real-world cybersecurity issues.
2. Key AI/ML techniques (such as decision trees, neural networks, and adversarial examples).
3. "How" the machine learns.
4. Data and its features.
5. Optimization (as learning).
6. Issues related to generalization (and overfitting/underfitting) in AI/ML models.
7. How bias can impact aspects (and phases) of ML.
8. Issues of ethics.
9. Adversarial Thinking i.e. whenever we discuss an ML model for classification/prediction, *also think of how it can be fooled or exploited.*

Data exploration with Common Online Data Analysis Platform (CODAP). CODAP empowers students to explore data to gain insight about underlying patterns. Using CODAP (Finzer 2016) from within NetsBlox allows students to leverage the complementary strengths of each; NetsBlox is used to preprocess or filter segments of the dataset while CODAP is used to gain insight into the dataset as well as select regions of interest for further exploration (Figure 1). In the context of ML, this is powerful when building rule-based classifiers or decision trees.

Pre-programmed games and interactive explorations. Pre-programmed games are used to introduce fundamental ML topics in an interactive, engaging way prior to programming activities. As these games are programmed in NetsBlox, students interested in learning more can easily probe deeper into the application as it readily accessible in a familiar programming language. Along with pre-programmed games, a variety of interactive web-based tools and visualizations are used for introducing concepts and exploring applications with real-world datasets. Pre-programmmed games include the Find the Minimum Game (described below) and a Registration Activity for collecting human biometric data during account creation for a mock website. Interactive explorations include Adversarial JS (shown in Figure 7) and GAN Lab (Kahng et al. 2018). Unlike other curricula which may culminate in the use of some of the interactive explorations, these are used in tandem with hands-on interactions with the underlying algorithms to understand the concept or explore the concept through alternative tools.

Leveraging research on multiple levels of abstraction, Parson's Problems and Subgoals for engaging with ML models. In order to help learners at all levels of interest and ability succeed in engaging with non-trivial ML algorithms, we employ "levels of abstraction" (Csizmadia, Standl, and Waite 2019; Waite et al. 2017) as a scaffolding tool. We introduce the basic algorithm in pseudocode, then provide "subgoal"-inspired (Morrison, Margulieux, and Guzdial 2015) blocks for implementing the algorithm and Parson's problems (Denny, Luxton-Reilly, and Simon 2008; Parsons and Haden 2006) for code completion that have been shown to scaffold introductory programming (Ericson, Margulieux, and Rick 2017). At the lowest level, we provide the entire code that implements the algorithm. The level(s) provide teachers with options to help learners engage at varying levels, while still ensuring that they all leave with an intuition of how the ML technique works.

ACT Activities Sequence

Guided by the goals and big ideas described in the previous section, we started our work by identifying real world, topical cybersecurity issues such as bots or Denial of Service (DoS) attacks, or phishing (along with related articles or videos to serve as "hooks" in a learning setting). We then conceptualized programming activities and projects that mapped to key AI/ML concepts and our target big ideas (Table 1). This mapping was refined and put into a coherent sequence as activities were iteratively designed. The following sections describe the ACT sequence of activities and NetsBlox-based programming projects.

Rule-Based AI to Mitigate Denial of Service Attacks Denial of Service attacks are a simple vulnerability of the naive chat application (involving a client and server chat app) built while introducing NetsBlox. Not only are they a natural extension of the initial NetsBlox activity, but they are also common in the real-world with many stories of effective attacks and mitigations. In this activity, we explore different techniques for mitigating denial of service attacks and discuss how this can be viewed as classification through a machine learning lens. That is, if we knew what requests were from legitimate users and which were from malicious users, denial of service mitigation would be simple!

Text Classification & Cyberbullying After introducing the concept of classification with the chat application, we extend it further to add content moderation using AI. Using the `ParallelDots` service from NetsBlox, students can easily leverage natural language processing (NLP) models such as those for classifying abusive text (shown in Figure 1b). This activity gives students a bit of freedom to explore how they might use the NLP tools to add content moderation capabilities. For example, students may choose to disable abusive text being sent from their client as well as filter out harmful text received from other students. As with most of the activities, this activity is designed to have many opportunities for customization and for the students to really personalize the project as they see fit. For example, color-coding text based on detected sentiment or sarcasm are a couple

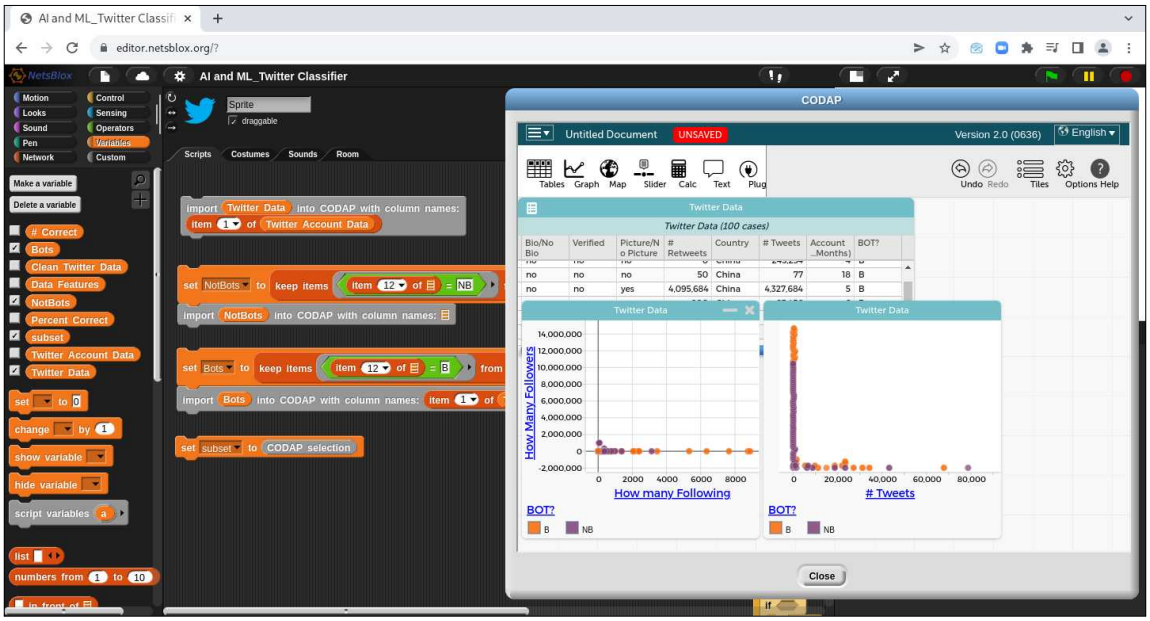

Figure 2. Dynamic data exploration with CODAP within NetsBlox. The data explorer window has multiple plots showing different features and their correlation with the label of each point. NetsBlox scripts filter the dataset and retrieve the current selection from the plots.

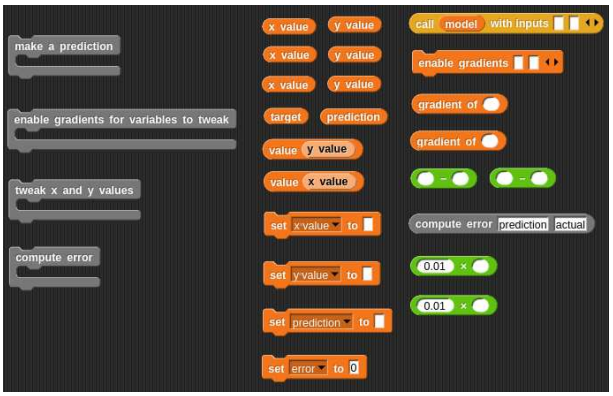

Figure 3. "Layers of Abstraction": Students can explore the gray blocks on the left that represent subgoals for the algorithm and are implemented so students get feedback on a correct/incorrect sequence. Interested students can also use the blocks presented on the right as a Parson's problem to implement each subgoal.

easy extensions which students could implement with relatively little difficulty. Although they are not yet creating their own models, this activity allows students to gain more familiarity with fundamental machine learning concepts while using an existing pre-trained model.

Twitter Bot Classification & Decision Tree Building The first ML algorithm that we implement is aimed at building a decision tree. This activity begins with exploring a synthetic Twitter account dataset using CODAP (Finzer 2016). Dynamic data exploration helps the students examine features in a dataset and gain hands-on experience with concepts like linear separability. Easily separable classes, like the orange class shown in Fig. 2, provide an opportunity to teach some fundamental ML concepts such as feature selection and classification. More challenging classes enable us to introduce probabilistic concepts such as model confidence in classification. Exploration of possible features around which the data can be classified allow students to then build a classifier using techniques such as decision trees.

After the CODAP exploration, students first implement a simple set of rules (composed largely of just `if` statements) to predict if an account is a human or bot. During the process, students iteratively explore how different features are related to the labels i.e., "bot" or "not bot" of the data and then choose a way to split the data that seems to best separate the bots from the human accounts.

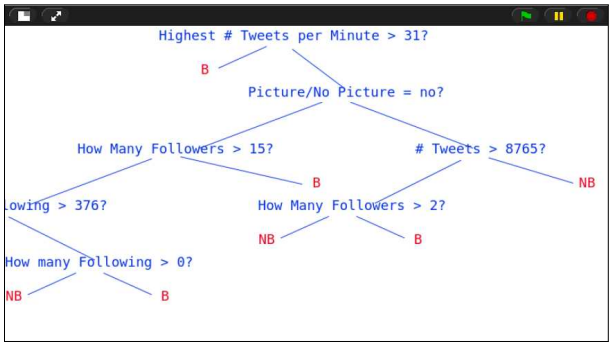

Figure 4. Exploring a decision tree learned from a Twitter bot dataset.

Programming Project	AI/ML Concepts
Project 1: Build a text-msg (chat). Intro to NetsBlox	
Project 2: DoS example & rule-based AI systems	Rule-based AI systems; classification
Project 3: Sentimental writer (also introduces RPC block) or Moderated Chat Client	Use of Parallel Dots NLP API to classify chat text as abusive or not (or neutral)
Project 4a: Twitter bot classification (start with CODAP data exploration) **Project 4b**: Phishing data - building intuitions about data/features/classification	Decision trees, classification, intuition about data and its features & labels
Project 5a: Introduce registration bot activity. Create your own registration bot! **Project 5b**: Play the Gradient Descent game; Engage in abstrations of GD algorithm; Train a bot detection model using GD.	Neural networks; optimization (as learning); gradient descent
Project 6: Explore how we can fool the bot detection model by generating adversarial examples. Generate adversarial examples using AdversarialJS online.	Adversarial examples; optimization
Project 7: What if we create a model to make adversarial examples? Circle GA(N) exercise; GAN lab activity	Realistic content generation via AI; GANs; optimization

Table 1. ACT NetsBlox programming activities sequence along with targeted AI topics.

After creating a rule-based classifier, we reflect on how these sets of `if` statements could be viewed as a decision tree. In a metacognitive step, we reflect on our process of manually creating rules to classify the data and consider how this could be automated. Students then formalize it with pseudocode, and then complete a decision tree building algorithm presented as a Parson's problem—first at a higher level of abstraction where they use subgoals instead of primitive blocks, and then implement the subgoals in code.

Adversarial thinking is a useful skill in computing, including in cybersecurity and AI/ML (Young and Krishnamurthi 2021). Thinking about how models can be fooled through adversarial examples is one rich application of this skill which challenges the students to critically examine the ML model for vulnerabilities. Interpretable models, like decision trees, provide an easy introduction to the topic. Building intuition through hands-on experience can be relatively simple. We use the visualization of the classifier (Fig. 4) to extend the Twitter Bot classification exercise with a follow-up one: Can you construct an example that is classified as "not bot"? It is an organic next step to investigate how to change an existing bot to be misclassified.

Finally, students then apply their understanding from the Twitter bot activity to a phishing dataset, starting with examining features of phishing vs non-phishing emails and ending with use the decision tree model coded earlier to classify phishing emails.

Optimization & Find the Minimum Game Students must understand that the "learning" problem in machine *learning* is essentially an **optimization** problem, where an objective, fitness, or error function is defined and the goal of the algorithm is to either maximize or minimize the given function. Students of this age/ability are not in a position to code complex optimization, but we believe they ***can*** develop intuitions through carefully selected activities, games, or code examples that bring home the fundamental ways in which optimization works. For example, if abstractions are presented to make gradients accessible without requiring mastery of calculus, gradient descent is a relatively simple concept that can be made accessible to early teen students.

We designed an interactive game, "Find the Minimum Game" (Fig 5), to introduce the idea of learning through minimizing error (optimization) and build an intuition of **gradient descent** (Ruder 2016), an algorithm often used in neural networks. Students embody an optimizer and try to find the minimum of an unknown/invisible function/graph. At every click on the blank screen, an arrow at the mouse x-coordinate and y-value of the function shows the slope of the function and directs the player toward the minimum.

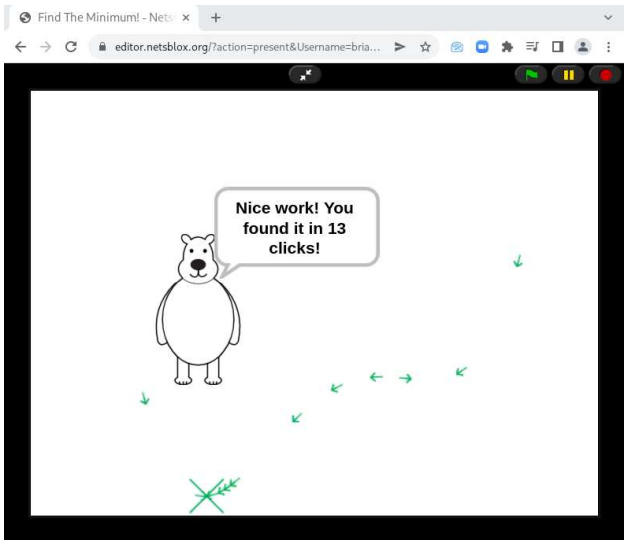

Figure 5. "Find the Minimum" game helps develop preliminary intuition of how the gradient descent algorithm works.

Registration Bot Detection with Gradient Descent This activity uses detection of a "registration bot" (based on mouse movement data) as a cybersecurity context for using gradient descent. Given mouse movement data of humans and bots while registering for a mock website (generated from an earlier activity), students create a ML model to classify a registration as performed by a human or a bot. First, we explore the dataset and choose a way to represent the data such as its length and average speed. Next, we come up with a simple way to classify the data points where we just learn an "importance number" (or weight) for each of the features and then combine them to get the prediction. Finally, we connect this with the earlier "Find the Minimum" activity and formalize the training algorithm through pseudocode, subgoals, then a Parson's problem.

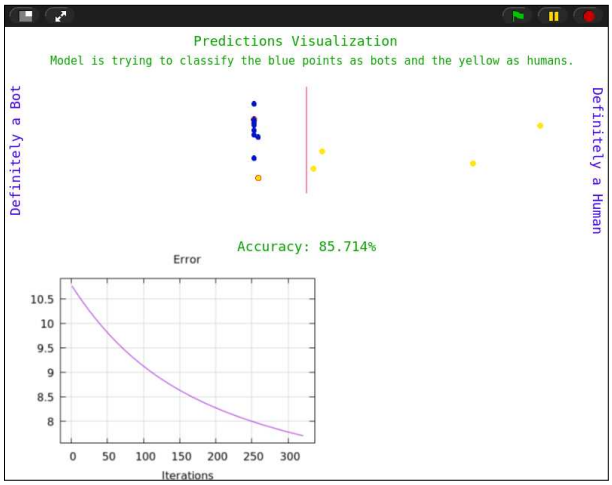

Figure 6. Visualization of model performance while training registration bot classifier.

Unlike projects in many existing AI/ML curricula, this activity introduces students to the entire lifecycle of designing an ML model. Students are thus able to not only build intuition through hands-on experience with ML but also explore cybersecurity aspects more deeply. For example, students may try to improve their registration bot to fool the model. Alternatively, students can perform a data poisoning attack and try to generate human data that behaves more like a bot to make the model misclassify their current bot. These types of extensions both exercise students' adversarial thinking and help them build a deeper intuition about ML concepts themselves. Additionally, understanding the relationship between optimization and machine learning empowers students to think critically about AI and bias in ML models (see below), and, importantly, avoid falling prey to widespread anthropomorphization of ML and AI.

Generative Adversarial Models Building on both the earlier themes of optimization and adversarial thinking, we introduce generative adversarial networks (GANs)[1]. This

[1]We use simpler models than neural networks so they are technically not generative adversarial *networks*.

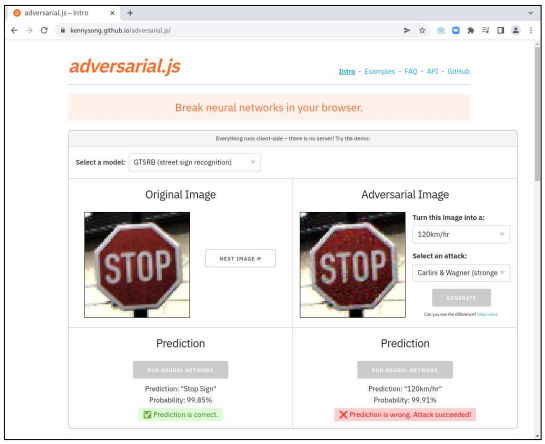

Figure 7. Students explore adversarial examples with real-world datasets in Adversarial JS before implementing them.

activity starts by building on the adversarial examples created using the registration bot detection activity. However, rather than creating a single point to fool the model, students are asked if we could train a model to automatically create point(s) to fool the model. Following the approach used in the prior activities, we frame the task as an optimization problem by deciding what we would like to minimize and what parameters we are changing to do so. When only considering the generator, this is almost identical to the original gradient descent problem except we are trying to guess the "incorrect" label given by the discriminator.

As in the earlier activities, we start with the "big idea" approach and then discuss how this might look in pseudocode. After completing the pseudocode as a class, students are tasked with implementing the training of the generator as a Parson's problem using subgoal blocks. Although not likely suitable for every class, we also have a Parson's problem for the low-level blocks in which students implement each of the subgoals with the provided blocks.

After completing the training algorithm for the generator, students have a functional example of training a GAN in the browser. The completed project is shown in Figure 8. In this example, the user has created the positive examples in the lower left. The generator was randomly initialized and created the negative examples in the top right. The circle is a visualization of the discriminator's predictions and is effectively circling only the positive points. As the generator is trained, the synthetic points in the top right will move toward the center of the circle.

Generalization & Over/Underfitting

The ability of a model to generalize to unseen points is critical when training ML models and is complementary to adversarial examples. Models that generalize poorly are often easy to fool. This is no surprise after exploring the learned decision tree (as shown above); the individual parameters that have been learned may seem somewhat arbitrary. Enabling students to train both interpretable and black-box ML models empowers them to gain hands-on experience with

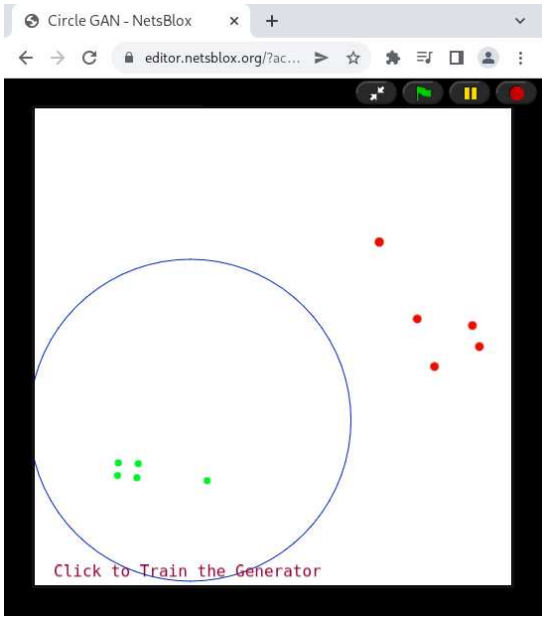

Figure 8. Circle GAN activity: a classifier is trained to circle only the green points (created by the user). The generator creates the red points and is trained to fool the classifier.

over/underfitting as well as investigate the impact of different data sampling approaches on the resultant models.

Understanding Bias and Critical Interrogation of the Impacts of ML Models.

Although AI/ML algorithms are not *inherently* biased, bias in a dataset or decisions related to feature selection or optimization can have serious consequences when an ML model is put to use in decision-making that impacts people and real-world situations. As the usage of AI/ML models becomes increasingly ubiquitous, the impact of bias and understanding of what a model has in fact learned has become increasingly important. Building intuition through hands-on experiences that "lift the hood" on ML as our activities do, enables students to gain a deeper understanding about the impact of the dataset and objective.

When combined with the other key ideas listed above, students are able to have deeper insight about potential causes and impacts of bias. For example, the first key idea facilitates simple early questions about the features used to represent the data points. Learning about generalization can be a catalyst for interrogating the origins of a dataset. How might that impact the under- or overrepresentation of various types of data in the dataset? How will this affect generalization? Viewing machine learning as an optimization problem raises questions about the quantity that is being optimized. How does this compare to the way the model is going to be used when deployed? If we are training the model to predict on historical data, are we sure that the past data is something we want to try to replicate? What if there were social or cultural issues that disenfranchised some demographic?

Teacher Professional Development+Feedback

Pilot #1. We first conducted a 15 hour pilot online teacher workshop over a period of 4 sessions in Fall 2021 with 7 high school teachers from across the US, with the stated goal of gathering feedback. 6 teachers completed the pre-post survey. Post-survey responses suggested that teachers found the pilot activities intertwining AI and cybersecurity to be suitable, innovative, and helpful for their own learning. Based on teacher feedback, activities were refined to add more levels of scaffolding.

The mean rating of activities on the innovativeness of the ACT curriculum (1=Not at all innovative; 5=Very innovative) was 4.5. Mean teacher rating on the appropriateness of the intertwining of AI & Cybersecurity in the activities (1=Not at all connected/Does not make sense; 5=Very well connected/Makes sense) was 4.3. Mean teacher comfort level with AI topics changed from 2.7 to 3.7 (1=Not at all familiar (it's totally new to me); 5= I'm an expert (I teach it to middle/high school or older students)). In rating specific ACT activities with choices: *Not a great activity (least favorite)*; *Good idea but needs a lot of improvement*; *Almost there- good activity/experience*;*Great Activity (among my faves)*; *Did not attend session*, responses to most activities was mostly "Almost there" or "Great Activity" Two or more teachers indicated that message passing (chat) & DoS, CyberBullying, Decision Tree/TwitterBot, Gradient Descent, were a "Great Activity (among my faves)". The most popular was the cyberbullying/sentiment analysis activity with 4 teachers marking that as "Great"; next was the TwitterBot/Phishing/Decision Tree activity with 3 teachers marking that as "Great Activity".

Pilot #2. We then recruited 7 teachers for a week-long online summer workshop. Once again, the overall the response to the second workshop was overwhelmingly positive. In response to the question: *How important is it to understand AI and machine learning as it is related to cybersecurity?* 4 teachers marked ***extremely important*** and 1 teacher, ***very important***. They hailed the curriculum as innovative and that it did a great job of intertwining AI and cybersecurity.

Conclusion & Next Steps

In this work, we explore how we can lift the hood on fundamental ML concepts for high school students with minimal required prerequisites. This includes the use of a block-based programming environment, Parson's problems, levels of abstraction, carefully designed abstractions, and subgoals to make the activities more accessible. Through supporting a rich interaction with the concepts through hands-on experiences and programming, we hope to help students better develop intuition about the core ML concepts which can then be applied to both other ML contexts and discussions about the use and challenges of AI/ML in the real-world. Our next steps include continuing to develop the activities including differentiation for students of different interests and abilities. We plan to pilot this in classes in Fall 2022 and will use the feedback to continue to refine the activities and curriculum.

Acknowledgements

This material is based upon work supported by the National Science Foundation under grant number 2113803. We are grateful to Derek Babb for his contributions to this project.

References

Abelson, H.; and DiSessa, A. 1986. *Turtle geometry: The computer as a medium for exploring mathematics*. MIT press.

Alvarez, L.; Gransbury, I.; Cateté, V.; Barnes, T.; Ledéczi, ; and Grover, S. 2022. A Socially Relevant Focused AI Curriculum Designed for Female High School Students. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11): 12698–12705.

Bransford, J. D.; Brown, A. L.; Cocking, R. R.; et al. 2000. *How people learn*, volume 11. Washington, DC: National academy press.

Broll, B.; Lédeczi, A.; Stein, G.; Jean, D.; Brady, C.; Grover, S.; Catete, V.; and Barnes, T. 2021. Removing the Walls Around Visual Educational Programming Environments. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 1–9. IEEE.

Broll, B.; Lédeczi, A.; Volgyesi, P.; Sallai, J.; Maroti, M.; Carrillo, A.; Weeden-Wright, S. L.; Vanags, C.; Swartz, J. D.; and Lu, M. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*, 81–86.

Brown, J. S.; Collins, A.; and Duguid, P. 1989. Situated cognition and the culture of learning. *Educational researcher*, 18(1): 32–42.

Bruner, J. S. 1960. *The process of education*. Harvard University Press.

Burgsteiner, H.; Kandlhofer, M.; and Steinbauer, G. 2016. Irobot: Teaching the basics of artificial intelligence in high schools. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Csizmadia, A.; Standl, B.; and Waite, J. 2019. Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education*, 18(1): 41–67.

Denny, P.; Luxton-Reilly, A.; and Simon, B. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*, 113–124.

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.

Druga, S.; Otero, N.; and Ko, A. J. 2022. The Landscape of Teaching Resources for AI Education. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1*, ITiCSE '22, 96–102. New York, NY, USA: Association for Computing Machinery. ISBN 9781450392013.

Druga, S.; Vu, S. T.; Likhith, E.; and Qiu, T. 2019. Inclusive AI literacy for kids around the world. In *Proceedings of FabLearn 2019*, 104–111.

Ericson, B. J.; Margulieux, L. E.; and Rick, J. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*, 20–29.

Evangelista, I.; Blesio, G.; and Benatti, E. 2018. Why are we not teaching machine learning at high school? A proposal. In *2018 World Engineering Education Forum-Global Engineering Deans Council (WEEF-GEDC)*, 1–6. IEEE.

Finzer, W. 2016. Common online data analysis platform (CODAP). *Emeryville, CA: The Concord Consortium.[Online: concord. org/codap]*.

Gainsburg, J. 2008. Real-world connections in secondary mathematics teaching. *Journal of Mathematics Teacher Education*, 11(3): 199–219.

Glazewski, K.; Ottenbreit-Leftwich, A.; Jantaraweragul, K.; Jeon, M.; Hmelo-Silver, C.; Scribner, J. A.; Lee, S.; Mott, B.; and Lester, J. 2022. PrimaryAI: Co-Designing Immersive Problem-Based Learning for Upper Elementary Student Learning of AI Concepts and Practices. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2*, 628–628.

Guerreiro-Santalla, S.; Mallo, A.; Baamonde, T.; and Bellas, F. 2022. Smartphone-Based Game Development to Introduce K12 Students in Applied Artificial Intelligence. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11): 12758–12765.

Harvey, B.; and Mönig, J. 2017. Snap! reference manual. *URL http://snap. berkeley. edu/SnapManual. pdf*.

Hitron, T.; Orlev, Y.; Wald, I.; Shamir, A.; Erel, H.; and Zuckerman, O. 2019. Can children understand machine learning concepts? The effect of uncovering black boxes. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, 1–11.

Jatzlau, S.; Michaeli, T.; Seegerer, S.; and Romeike, R. 2019. It's not magic after all–machine learning in snap! using reinforcement learning. In *2019 IEEE blocks and beyond workshop (B&B)*, 37–41. IEEE.

Kahn, K.; Megasari, R.; Piantari, E.; and Junaeti, E. 2018. AI Programming by Children using Snap! Block Programming in a Developing Country. In *European Conference on Technology Enhanced Learning*.

Kahn, K.; and Winters, N. 2017. Child-friendly programming interfaces to AI cloud services. In *European Conference on Technology Enhanced Learning*, 566–570. Springer.

Kahn, K.; and Winters, N. 2021. Learning by enhancing half-baked AI projects. *KI-Künstliche Intelligenz*, 35(2): 201–205.

Kahng, M.; Thorat, N.; Chau, D. H.; Viégas, F. B.; and Wattenberg, M. 2018. Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE transactions on visualization and computer graphics*, 25(1): 310–320.

Lane, D. 2021. *Machine learning for kids: A project-based introduction to artificial intelligence*. No Starch Press.

Lee, I.; Ali, S.; Zhang, H.; DiPaola, D.; and Breazeal, C. 2021. Developing Middle School Students' AI Literacy. In *Proceedings of the 52nd ACM technical symposium on computer science education*, 191–197.

Morrison, B. B.; Margulieux, L. E.; and Guzdial, M. 2015. Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, 21–29. New York, NY, USA: Association for Computing Machinery. ISBN 9781450336307.

Parsons, D.; and Haden, P. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*, 157–163.

Payne, B. H. 2019. An ethics of artificial intelligence curriculum for middle school students. *MIT Media Lab Personal Robots Group. Retrieved Oct*, 10: 2019.

Royal Society (Great Britain). 2017. *Machine Learning: The Power and Promise of Computers That Learn by Example*. Royal Society. ISBN 9781782522591.

Ruder, S. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Thormundsson, B. 2022. Virtual Assistant Technology - Statistics Facts. https://www.statista.com/topics/5572/virtual-assistants/. Accessed: 2022-09-30.

Touretzky, D.; Gardner-McCune, C.; Martin, F.; and Seehorn, D. 2019a. 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.

Touretzky, D. S.; Gardner-McCune, C.; Martin, F.; and Seehorn, D. 2019b. K-12 guidelines for artificial intelligence: what students should know. In *Proc. of the ISTE Conference*.

Van Brummelen, J.; and Lin, P. 2020. Engaging teachers to co-design integrated AI curriculum for K-12 classrooms. *arXiv preprint arXiv:2009.11100*.

Waite, J.; Curzon, P.; Marsh, W.; and Sentance, S. 2017. K-5 Teachers' Uses of Levels of Abstraction Focusing on Design. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, 115–116.

Young, N.; and Krishnamurthi, S. 2021. Early Post-Secondary Student Performance of Adversarial Thinking. In *Proceedings of the 17th ACM Conference on International Computing Education Research*, 213–224.