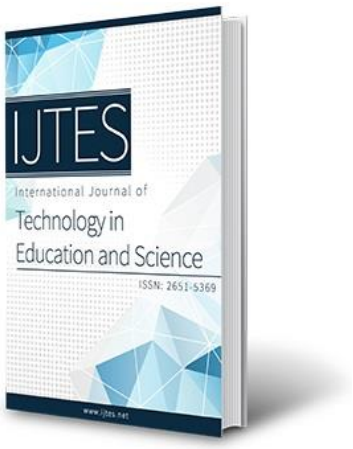

www.ijtes.net

Frameworks and Challenges for Implementing Machine Learning Curriculum in Secondary Education

Fletcher Wadsworth
University of Wyoming, USA

Josh Blaney
University of Wyoming, USA

Matthew Springsteen
University of Wyoming, USA

Bruce Coburn
University of Wyoming, USA

Nischal Khanal
University of Wyoming, USA

Tessa Rodgers
University of Wyoming, USA

Chase Livingston
University of Wyoming, USA

Suresh Muknahallipatna
University of Wyoming, USA

To cite this article:

Wadsworth, F., Blaney, J., Springsteen, M., Coburn, B., Khanal, N., Rodgers, T., Livingston, C., & Muknahallipatna, S. (2024). Frameworks and challenges for implementing machine learning curriculum in secondary education. *International Journal of Technology in Education and Science (IJTES), 8*(1), 164-181. https://doi.org/10.46328/ijtes.531

International Journal of Technology in Education and Science
2024, Vol. 8, No. 1, 164-181 https://doi.org/10.46328/ijtes.531

Frameworks and Challenges for Implementing Machine Learning Curriculum in Secondary Education

Fletcher Wadsworth, Josh Blaney, Matthew Springsteen, Bruce Coburn, Nischal Khanal, Tessa Rodgers, Chase Livingston, Suresh Muknahallipatna

Article Info

Article History

Received:
08 August 2023
Accepted:
14 November 2023

Keywords

Machine learning
STEM
Pedagogy
Coding
Education

Abstract

Artificial Intelligence (AI) and, more specifically, Machine Learning (ML) methodologies have successfully tailored commercial applications for decades. However, the recent profound success of large language models like ChatGPT and the enormous subsequent funding from governments and investors have positioned ML to emerge as a paradigm-shifting technology across numerous domains in the coming years. To cultivate a competent workforce and prepare students for success in this new AI-focused evolving world, the integration of ML is proposed to begin in compulsory education rather than in college courses or expensive boot camps. Unfortunately, ML is a complex and intimidating topic for high school teachers to engage with, let alone high school students. Based on our experiences hosting Machine Learning for High School Teachers (ML4HST) workshops for teachers teaching ML topics at our institution, we present in this paper various considerations for educating educators on the topic of ML. In particular, we discuss (a) overarching pedagogic strategies, (b) accessibility of resources such as computational hardware and datasets, (c) balancing theory and implementation, (d) appropriate selection of topics and activities for fostering understanding and engagement, and perhaps most importantly, (e) a compilation of pitfalls to avoid. Synthesizing these insights, we propose a framework for successfully empowering educators to introduce ML in the classroom.

Introduction

Machine Learning

Artificial intelligence, the discipline of empowering machines or software to behave intelligently, began as a field of study in 1956. Soon after, in 1959, the term machine learning was coined by Arthur Samuel at IBM (Samuel, 1959). Currently, ML is a subdiscipline of AI, specifically referring to the development of algorithms in a new way; instead of designing algorithms to transform data by hand, which is often time-consuming and infeasible for experts, the ML approach allows data to guide the learning of an algorithm or mathematical transformation. Alpaydin (2020) describes ML as follows:

> Roughly speaking, our approach starts from a very general model with many parameters, and that general model can do all sorts of tasks depending on how its parameters are set. Learning corresponds to adjusting

> the values of those parameters so that the model matches best with the data it sees during training. Based on this training data, the general model, through a particular setting of its parameters, becomes specialized to the particular task that underlies the data. That version of the model we get after training, that particular instantiation of the general template, becomes the algorithm for that task (p. 1).

As it turns out, these general models with many parameters come in many varieties, and certain model architectures have proven to be sufficiently general, powerful, and scalable to allow ML to tackle broader and deeper problems. In particular, the artificial neural network (ANN or NN) is a methodology that has developed over decades into the field of Deep Learning (DL). The simplest kind of ANN is a linear network, which implements a linear regression or method of least squares, used by Legendre and Gauss in the late 18th and early 19th centuries. However, the first network, which resembles a modern deep learning network, comprised of multiple layers of neurons (deep NN or DNN) and trained by stochastic gradient descent, was proposed by Shun-Ichi Amari in 1967 (Schmidhuber, 2022). After the 1982 proposal of training neural networks efficiently with backpropagation, the basis for an ML model possessing the generalizability and predictive power to be widely applied to computational tasks was formed.

Since then, computational power has been the driving factor in the continued integration and development of ML models and applications. In recent years, adapting the Graphics Processing Unit (GPU) to perform highly parallel computations allowed deep learning research to accelerate dramatically, allowing faster training and, thus, bigger and more expressive models. After NVIDIA released Compute Unified Device Architecture (CUDA) as a C language extension for general-purpose GPU programming in 2007 and the GPU-accelerated library of deep neural network primitive operations, called CuDNN, in 2014 (Pandey et al., 2022), highly optimized deep learning routines were available to practitioners, and efficacy of deep learning models exploded. Although the basic neural network still uses DNN model architectures and training, many variations have been developed. Data structuring and model operation changes have allowed networks to leverage the properties of different applications and data types more effectively. Some key examples of model and training development are described in Table 1.

Table 1: Notable Deep Learning Model Developments

Model	**Architecture Changes**	**Application**
Convolutional Neural Network (CNN)	Convolution with kernels instead of weights	Data with grid-like topology, e.g. image data
Recurrent Neural Network (RNN)	Hidden state shared between steps of sequences in data	Data with sequential dependencies, e.g., time series data
Autoencoders	Encoding and decoding learn efficient representations of data	Dimensionality reduction, feature learning
Generative Adversarial Network (GAN)	The game between the generator and discriminator network	Image generation, data augmentation
Transformer with self-attention	Assign weight to different elements of the input sequence	Natural language processing, large language models (GPT)

Many popular ML algorithms do not fall under the umbrella of deep learning, which is still used today. Examples include decision trees, support vector machines, Bayesian networks, etc. These approaches are generally simpler to describe, train, and interpret than cutting-edge deep learning models. However, our focus is primarily on deep learning due to its comparatively high predictive power and the continuing dominance of deep learning models in industry and research. With all breakthrough ML technologies primarily driven by deep learning techniques, we feel it is prudent to focus on the more powerful and commercially successful domain of deep learning when deciding what topics to cover when educating our future workforce.

Education

As quickly as AI develops as a field and the ever-changing landscape of ML theory and application progresses, it isn't easy to pin down what ML education in K-12 consists of beyond short-lived trends and their broad contours. With the landmark achievements in ML that supersede the results of classical AI rule-based systems, initiatives, including the more mathematically intensive ML models, have been filtered into AI education. Marques et al. (2020) identified 30 initiatives focusing on ML basics and neural networks for K-12 cohorts. Specifically, at the high school level, efforts to clarify the key topics of ML in the curriculum have been made by various researchers (Yu & Chen, 2018).

As ML model complexity and performance have increased, so too have the toolkits that empower amateur programmers, domain experts, and educators to more easily access or cultivate datasets and implement sophisticated models without a deep understanding of the underlying computations. Some ML applications may require complex or custom workflows, allowing scripting language libraries such as PyTorch, scikit-learn, and others to abstract much of the underlying operations. For more classical applications, many projects have further hidden the complexity of the model architecture and training algorithms, such as block programming (Estevez et al., 2019) and web-based ML platforms, including but not limited to those used in our workshop (see Appendix A).

One group characterized the five “big ideas” in AI for childhood education: computer perception through sensors, representing the world with models, computers learning from data, making AI properly interact with humans, and positive and negative influences of AI in society (Touretzky et al., 2019). There is a clear tradeoff when educating broadly about a set of technologies as impactful as AI/ML; with limited time, an educator must decide between depth and breadth, focusing on the theory and implementation of ML models or speaking to the societal impacts of human-computer interaction and ethical concerns regarding AI systems. Our expertise lends us to concentrate on the technical aspects of ML and leave the social aspects to others more qualified, but we would be remiss not to emphasize its importance. The modern citizen is inundated by decision-making machines, from image hosting platforms recognizing the people in one’s photos to streaming services and social media outlets recommending content based on what a user will likely engage with. In understanding the mechanisms underlying these opaque and unnerving prediction machines from a young age, these technologies can be demystified to educated young adults, and their harmful effects on the individual’s privacy and autonomy can be curtailed.

Method

Workshop

In the summers of 2021, 2022, and 2023, we hosted Machine Learning for High School Teachers (ML4HST), an interactive workshop intended to broadly educate high school teachers in concepts of machine learning, with a pronounced focus on deep learning. The ML portion of the workshop totaled four days of demonstration and activities, which will be detailed here:

Day 1

- Introduction to AI and ML - introductory lecture to definitions, examples, and paradigms of AI and ML
- Online ML activities – Quick, Draw! by Google, Scroobly, Thing Translator, Lobe AI
- ML fundamentals – lecture providing ML and DL terminology and explaining the broad contours of datasets, learning, and neural networks.
- Introduction to Perceptron – lecture providing the mathematical formulation of the artificial neuron
- Rock, Paper, Scissors – training classifier with Lobe AI for gesture recognition and playing it on Raspberry Pi
- Iris/Animal classifier with scikit-learn – interactive demonstration on classification tasks using scikit-learn logistic regression and support vector machine models
- Temperature trends –interactive demonstration for polynomial regression with scikit-learn
- Handwritten digit classification – interactive demonstration for image classification with multilayer perceptron model in scikit-learn

Day 2

- Face tracking – interactive demo for object detection with Haar cascade
- ChatGPT – interactive lecture on ChatGPT capabilities, limitations, and use in the classroom
- Intro to PyTorch – interactive lecture on basic model building and training with PyTorch
- Bird sound identification – interactive demo for a pre-trained RNN to identify bird calls
- PyTorch Lightning and Copilot – interactive demo on using lightning and copilot to increase ease of programming models and training algorithms.
- Vehicle detection – interactive demo on transfer learning with R-CNN to detect vehicles in videos with PyTorch

Day 3

- Melody harmonization – interactive demo on transformer model to generate harmonies using PyTorch

- Plant identification – interactive demo on transfer learning to identify 50+ plant species using PyTorch
- Introduction to CNNs – lecture with demonstration of convolutional neural networks
- Advanced ML architectures – interactive lecture on RNN and GAN models with PyTorch
- TurboPi – face tracking with Raspberry Pi-based mobile platform

Day 4

- TurboPi – line following with Raspberry Pi-based mobile platform
- Obstacle avoidance – CNN classifier to avoid objects with DJI Tello EDU drone
- JetHexa – Gesture tracking and robot maneuvering with NVIDIA Jetson-based hexapod mobile platform
- PuppyPi – CNN classifier for ball following with Raspberry Pi-based quadruped mobile platform

There were 14 participants in the workshop: 10 high school educators, three accompanying students, and one professor in the field of engineering with no ML experience. The workshop was all in person at the University of Wyoming. The modules were presented in a computer lab, with each participant accessing a computer running Windows 11, with an AMD Ryzen 7 5800X 8-core processor and NVIDIA RTX A6000 GPU. Other applied modules, including modules on robotic platforms, were conducted in laboratories with laptops remotely connected to the platform computer.

Observations

It is important to note that our workshop contained no assessment of participant comprehension and mastery of presented topics. We lack the counterfactual evidence to state whether the workshop produced a measurable effect on the effectiveness of these educators in teaching ML topics. Our results and discussion are informed primarily by the authors' assessment of the teachers' comprehension. Additionally, we released a questionnaire that asked several open-ended questions on the perceived efficacy of the workshop material and presentation; direct quotations from the feedback will be used to support claims as needed. See Appendix C for the list of feedback questions.

Pedagogic Strategies

Lecture

There is no obvious way to teach an inherently mathematical topic without some form of lecture; interactivity can be incorporated, yet unheard concepts must be explained. All workshop modules that introduced a new topic, be that a new model architecture or concept in ML, began with a lecture. As will be discussed, balancing theory and implementation is critical to keeping learners engaged, and effort was made to keep the lectures brief, stimulating, and at an appropriate level of depth.

Interactive Learning

The clearest observation we made throughout the workshop is that interactive lessons are preferred to lecture-type presentations. In the context of machine learning, interactive can be seen as a function of integrating data collection, labeling, and processing into the lesson, having participants write code for takeaway concepts themselves, and lesson dependent user interaction with model prediction.

Regarding data set curation, some but not all demonstrations presented in the workshop included a component of data collection (see Table 3) and exploration. Despite data collection generally being the most arduous step in the ML workflow, participants found the process valuable to understand ML as a fundamentally data-driven enterprise better. In the feedback questionnaire, one participant wrote, “I felt that by using pictures/videos collected by us, I had a vested interest in seeing what happened or what was being processed by the programs.”

Concerning students writing code themselves, this is a difficult balance to be struck. Too little programming leaves the ML process opaque and does not empower students to explore model creation and training led by their curiosity. On the contrary, expecting students to type many lines of code is futile; it is unlikely that even a small minority of ML beginners will have appreciable experience in programming to avoid unnecessary frustration in writing and debugging. Our interactive sessions had various amounts of exposure to the underlying scripts, and depending on the topic's complexity, variable success kept participants from becoming discouraged.

User interaction with the trained model is likely the most important consideration in devising educational strategies for ML; without demonstrating how trained model behaves with new data, there is scarcely a point in introductory ML education since ML is fundamentally a discipline of engineering solutions. By assessing topic selection and presentation with this in mind regarding our workshop, we conclude that integrating the trained model into an application is crucial for students to internalize the broad contours of ML. Some of our modules, particularly the classifier models, demonstrated the trained model by running predictions on some testing data and presenting statistics such as prediction accuracy. While this does demonstrate that a model has learned something from the data, it can fail to exhibit the purpose of developing the model in the first place. Why does one need a computer program to tell them that a picture is of a cat or a dog? Other modules demonstrated the trained model more vividly, such as detecting and localizing cars in videos collected by participants or allowing participants to generate harmonies to their music melodies. This approach firmly associated the unfortunately tedious ML workflow with an application that students can interact with; they can present their data to the model, investigate in what situation the model succeeds and fails, and strategize about improving its performance, all guided by their interests and curiosities.

Integrating data collection and labeling into an ML module depends entirely on the topic of interest and the method used. In ML, a data set is typically split into training and test data, where training data is presented to the model to adjust the model parameters and test data is used to assess the model’s predictivity on yet unseen data. For certain applications, assembling a training data set can be tedious due to the need for data labels (sorting data samples by class membership, annotating images with bounding boxes, etc.). For other applications, collecting

data is quite straightforward, e.g., collecting a short video of vehicles on the road for vehicle detection. It may be prudent for the training data to be assembled completely and allow students to collect a small amount of data to augment the training data or serve as test samples.

All but the simplest implementations of custom ML models and training loops require competency in programming tools and environments, particularly Python, and a considerable amount of time is needed for writing and debugging complex scripts. Appropriate exposure to the code is a critical question; asking teachers or students to implement and train a neural network from scratch is prohibitive, and hiding the entirety of the code does not empower students to explore or understand ML. Hiding boilerplate and tedious code while allowing access to key concepts in the training and testing scripts is the tradeoff we are trying to find the appropriate balance point.

It is not uncommon for lessons to focus too much on the details and fail to instill the overall context of the lesson. Students often lose the thread of a lesson and eventually question why they must learn the tedious specifics of a topic. This is a devastating trap in ML education. Machine learning is inherently complex, and a trained model returning prediction metrics on test data is only of interest to those who have intrinsic motivation to learn about ML. Thus, reconnecting the trained model to a demonstration of its predictions in an appropriate ML application is essential.

Computational and Data Resources

Due to the machines provided to our workshop participants, which possess capable and costly GPUs, our selection of topics, model architectures, and training parameters was not limited to the extent that educators using district-provided hardware would likely be. This was particularly evident in several of the workshop modules.

Bird sound identification used a large time-series classifier model (e.g., RNN) on high-fidelity sound data. Due to the large input data size and model complexity needed to parse it, available hardware would significantly impact training and prediction time. Additionally, such sequential models are less parallelizable and thus less amenable to speed-up with accelerators such as GPUs.

Vehicle detection employed R-CNN, a large pre-trained object detection model. When performing predictions on a single image, this model must pass a high-resolution image through a large convolutional network and region of interest proposal networks to generate an arbitrary list of predictions for bounding boxes for each object detected in the image. Fine-tuning only a single layer of weights on the workshop systems took around 30 minutes. While popular object detection models are available for download with pre-trained parameters, eliminating the need for coding the model and time-intensive training blocks, simple prediction on multiple images may require significant hardware resources rendering these models infeasible on non-specialized machines.

Data availability is critical for implementing ML models for tasks not amenable to self-curated data. Many data sets exist and are publicly available on websites like Kaggle; sets including student exam performance, customer

personality traits, face mask detection, popular song streaming, food choices, and others can be found and easily downloaded to use in educational settings. However, with every new data set comes the arduous process of wrangling the data structures to suit the needs of the particular lesson, which can require patience and expertise in data manipulation tools such as Python's *pandas* library. Workshop modules that used prebuilt public datasets are listed (see Table 2).

Table 2. Workshop Modules Using Public Datasets

Module	**Data sets**
Iris classifier	Kaggle - Tabular data on iris flower petal and sepal characteristics
Handwritten digit classifier PyTorch Lightning Advanced ML arch. (GANs)	MNIST - Handwritten Digits
Bird sound identification	Cornell - Bird sound files
Vehicle detection	KITTI – Object detection images (used for training)
Melody Harmonization	J.S.B. Chorales – symbolic music in array notation
Plant Identification	PlantNet-300k – Images of 1000+ plant species

The other option for data sets is for the educator and students to cultivate the data samples collaboratively. While this can add time and logistical obstacles to an ML project, it tends to be more engaging for students to have a role in project success. Data curation and cleaning are also crucial components of the typical ML workflow. Since the success of ML applications is largely determined by the size, quality, and variety of training data, ML education cannot be complete without understanding the approaches and challenges of collecting data samples. Workshop modules in which participants collected or augmented the training data are detailed in Table 3. All modules not mentioned in Tables 2 and 3 either contained no model training or leveraged an in-house dataset.

Table 3. Workshop Modules with Participant-Curated Data Sets

Module	**Data sets**	**Collection Time**
Online ML activities Intro to Lobe AI Rock Paper Scissors	Module dependent images	Captured during workshop via webcam
Polynomial regression	Twice daily temperature measurements	Recorded by participants before the workshop
Vehicle detection	Videos of traffic conditions (used for prediction only)	recorded by participants before the workshop
Melody Harmonization	Song requests (used for prediction only)	Sent by participants and encoded before the workshop
TurboPi	Images of line-following tracks	Collected during the workshop
Obstacle avoidance	Images with and without obstacles	Collected during the workshop
PuppyPi	Images with the ball in various positions	Collected during the workshop

Balancing Theory and Implementation

The modules we presented varied widely in the amount of ML theory presented and the degree to which each module resulted in a functional model. Some modules, including *ML fundamentals*, were entirely theoretical and intended to develop key mathematical concepts for M, such as model construction and optimization. In contrast, the robotic platform modules (TurboPi, object avoidance, etc.) focused on the practical implementation of previously discussed concepts, with participants actively collecting and labeling data, constructing and training models in Python, and integrating the applied models into application code to perform an intelligent task. However, most of the modules were a mix of theory and implementation. These modules introduced a new model architecture or ML paradigm, discussed the mathematical underpinnings at a level of depth constrained by the workshop schedule, and implemented an example model aimed at some engaging application.

Our workshop participants were high school educators, and they varied significantly in their comfort with math theory and programming. One might assume that teachers would prefer more application-centered demonstrations since these are more engaging, easier to replicate, and certainly easier to impart to high school students. However, participant feedback indicated that our participants intuitively understood the need for more challenging theoretical lectures and discussions. In response to the survey question, "How do you feel about the balance between ML theory (mathematics) and application (programming) in this workshop?" teachers responded:

- "I think it [the balance] was necessary and well done."
- "This year was a great balance of math and programming. I felt I have a much better understanding of what is happening."
- "…Understanding the mathematics behind the algorithms is a step towards explainable AI and a better spread of these applications in various industries."

Even the most basic ML algorithms, such as linear regression, require a reasonably sophisticated grasp of mathematical concepts, including linear algebra, calculus, and mathematical optimization, to fully understand how a machine learns decision rules are formed from data. Since these concepts are unlikely to be ubiquitously grasped within a cohort of high school students, balancing theory and practice becomes not unlike machine learning itself; there are no solutions, only tradeoffs. The exact placement of the fulcrum point in this balancing act can be challenging. Too much theory leaves learners overwhelmed with abstract and perplexing mathematics, which makes it difficult to internally reconnect with the overarching purpose of an ML project or lesson. Conversely, too little theory can result in superficial learning, where models are successfully implemented with little understanding of their internal mechanisms, capabilities, and limitations. Either extreme fails to precipitate a practical understanding of machine learning, or thus, care must be made to find the proper inflection point.

Appropriate Selection of Topics

All educators must contend that any unit on any topic must be forced into a finite amount of time. From this, it is clear that careful consideration must be made to choose the topics most pertinent to student understanding of the

topic broadly and to choose topics that are likely to be digestible and engaging. We chose to focus primarily on deep learning for the following reasons: DL models have higher predictive power than “classical” ML approaches, DL models are becoming easier to implement with the ongoing development of toolkits and programming libraries, and DL models are the de facto standard approach for the fascinating applications of ML, including intelligent tasks related to image, time series, and natural language processing.

Except for the iris classifier, animal classifier, temperature trends, and face tracking, all our workshop demonstrations leverage deep learning approaches to perform the task at hand. Since we covered only a small sample of ML topics, it isn't easy to make quantitative or qualitative observations about the effectiveness of one topic over another concerning participant engagement and understanding. We can state that no feedback, in conversation nor through the questionnaire responses, indicated that participants would have preferred to spend more time on classical ML algorithms. This aligns with our predictions since the limitations of these algorithms are evident even on toy datasets. In contrast, the superior predictive capabilities of DL models let machine learning amateurs approach more interesting problems and exceed performance expectations relative to the monotony of implementing the model. We feel that this greatly improves engagement and excitement about the possibilities of deep learning specifically. More about topic selection will be discussed in the next section.

Pitfalls

Perhaps the main mistake one could make in designing an educational module on ML is not selecting an appropriate depth with which to explore a topic. As previously mentioned, limiting the amount of mathematical theory in a lesson is crucial to retain student intrigue; equally important is discernment as to which topics are too abstruse to be distilled into a lesson intended for ML novices. The reverse also seems to be true, in that an ML topic must also surpass some unquantifiable floor of complexity to deliver sufficient results that make the lesson worth teaching.

Some of our workshop modules were just too ambitious, both from the conceptual understanding standpoint and workload volume. Take, for example, the melody harmonization demonstration, in which we used a transformer with a self-attention model framework to predict harmonic accompaniment to a given input melody. While participants appeared enthusiastic about the prospect of computer-generated music, the lesson fell flat in generating any interest and curiosity to under the model. Transformers and self-attention are novel and highly intricate models, and even a cursory overview of the theoretical background was too high for those lacking ML experience to engage. To add to the monotony, relating the encoding and decoding of music symbolically into an array format suitable for ML required further investment of time. Finally, to generate interest, participants were encouraged to submit song requests to test the trained model performance. Still, these melodies had to be hand-translated into the appropriate format. This task reduces the likelihood that a teacher without extensive musical and programming experience would wish to replicate such a lesson. These issues sum up to making a module that we predicted to be a success but was frustrating and boring. This module demonstrated an amusing topic but resulted in surface-level learning due to its difficulty and laborious implementation.

Another facet of difficulty is somewhat intrinsic to the practical reality of machine learning. Some models require large datasets and elite computational hardware to produce reasonable results on all but the most banal applications. This is more of an imbalance of difficulty and results, such that the outcome of training a complex model on a toy dataset generates uninteresting or flawed results due to the mismatch in model selection and available resources. Such an imbalance is exemplified in our workshop's advanced ML architectures module, particularly the exploration of generative adversarial networks (GANs). Training a GAN is somewhat of a game, where two deep networks are trained in tandem, with a generator that learns to transform an arbitrary noise input into a sample that resembles the training data and a discriminator that learns to discern real data samples from fake samples from the generator. The inherent difficulty in constrained optimization of two agents with different goals makes training GANs to produce reasonable fake data very challenging. Typically, a basic GAN performs poorly on any high-dimensional data set. In our workshop, we demonstrated GAN training on the MNIST handwritten digit data, consisting of low-resolution images of handwritten digits 0-9. The result was generated images that begin to resemble digits but are easily distinguishable and unimpressive. This illustrates that implementing a generative model like a GAN is too difficult to justify the mediocre results since improving the generated images requires larger models and more training time, which may be unavailable to a high school teacher. To summarize, classroom ML projects are ideally chosen with a balance in mind, which has a high ceiling and low floor for success, in that models are likely to learn interesting predictive or generative capabilities while avoiding unnecessary complexity. This may rule out some advanced ML paradigms as impractical to properly teach in a resource-constrained classroom.

ML projects must also produce results that interest the uninitiated to inspire interest in students. This is the main concern with classical ML techniques like basic regression models; they lack the predictive capabilities to impress someone without intrinsic motivation or other incentives to learn about ML. Despite the relatively straightforward implementation of such models, which might be appealing for ML beginners on a surface level, our participants did not express interest in learning more about these topics because they lack the predictive power to compel learners to do the difficult work of understanding foreign concepts.

A final pitfall that must be addressed is burying students too deeply into ML implementation mechanisms, primarily programming syntax and semantics. While a language like Python is readable and uncomplicated compared to other scripting alternatives, programming, and ML are ultimately disparate skill sets. Our approach was to create template code for the workshop participants, with much of the boilerplate code prewritten; students are only responsible for writing code corresponding to key ML concepts like model construction and training loops. We observed that the participants, some of whom had some background knowledge in Python, still expressed frustration at writing too much code without assistance. One survey respondent said, “When the code being added didn’t have [an] arrow to tell me what I was adding specifically, I was very lost.”

A given machine learning model, such as a multilayer perceptron network, is an apt metaphor for machine learning. At a high level, it has a relatively simple definition and abstract construction, underneath which lies a world of complexity that is impossible to comprehend concisely. Due to this complexity, it is difficult to partition ML into subcategories with members that are similar to each other and dissimilar to members of other categories,

not unlike trying to unravel the operations of a trained model to determine which input features correspond to which outputs.

Discussion

Pedagogic Strategies

Integrating data collection and labeling into an ML module depends entirely on the topic of interest and the method used. In ML, a data set is typically split into training and test data, where training data is presented to the model to adjust the model parameters and test data is used to assess the model's predictivity on yet unseen data. For certain applications, assembling a training data set can be tedious due to the need for data labels (sorting data samples by class membership, annotating images with bounding boxes, etc.). For other applications, collecting data is quite straightforward, e.g., collecting a short video of vehicles on the road for vehicle detection. It may be prudent for the training data to be assembled completely, allowing students to collect a small amount of data to augment the training data or serve as test samples.

All but the simplest implementations of custom ML models and training loops require competency in programming tools and environments, particularly Python, and a considerable amount of time is needed for writing and debugging complex scripts. Appropriate exposure to the code is a critical question; asking teachers or students to implement and train a neural network from scratch is prohibitive, and hiding the entirety of the code does not empower ML students to explore or understand. Hiding boilerplate and tedious code while allowing access to key concepts in the training and testing scripts is the tradeoff when we are trying to find the appropriate balance point.

It is not uncommon for lessons to focus too much on the details and fail to instill the overall context of the lesson. Students often lose the thread of a lesson and eventually question why they must learn the tedious specifics of a topic. This is a devastating trap in ML education. Machine learning is inherently complex, and a trained model returning prediction metrics on test data is only of interest to those who have intrinsic motivation to learn about ML. Thus, reconnecting the trained model to a demonstration of its predictions in an appropriate ML application is essential. This is quite evident in projects involving robotic platforms. Machine learning fits neatly into the environment-sensing component of robotic navigation so that students will more quickly grasp the role of ML in our world and experience feelings of ownership and investment in their engagement with these topics.

Computational and Data Resources

Due to the growing complexity of cutting-edge ML models, computational resources can become a primary constraint on teaching ML. As models become more capable due to their increasing number of parameters and operations, model prediction (i.e., forward pass of a single data sample) can have several seconds of latency, reducing the use cases of model application in a resource-limited setting. For example, a sophisticated object detection model like YOLO or R-CNN may be unable to perform and display inferences on a high frame-rate video in real-time on a low-cost classroom laptop such as an average Chromebook. Of course, model training is

much more computationally demanding than inference; with a sufficiently complex model, training time may be a lesson bottleneck akin to leaving a science experiment overnight to develop results, further complicating the logistical planning necessary to execute an ML unit.

It is also worth noting that, in addition to the multitude of online activities available for educators to showcase the capabilities of ML, there are also web-based resources for training custom models using cloud hardware resources. Google Colab is a cloud service that allows users to run Jupyter Notebooks (a user and education-oriented Python scripting interface) on freely accessible remote hardware. This and other similar services can raise the ceiling on possible ML projects by empowering educators to explore ML projects without concern about hardware limitations.

As discussed, many free and easily accessible data sets cover many application areas. These may be used for demonstrations of various ML topics. However, we feel that ML projects in the classroom should lean towards using data sets that are curated or at least augmented by students due to the sense of ownership of the project outcome and improved understanding of the role of data in a world increasingly driven by intelligent computers.

Balancing Theory and Implementation

This is an ever-present concern of STEM education, which we make no claims of cleanly delineating. ML is a mathematically intensive discipline; most high school students likely lack the math background and intrinsic motivation to engage deeply with these topics. Thus, these abstract concepts should be limited to the extent necessary to foster a high-level comprehension of the topic. A mathematical baseline is, however, required to avoid superficial learning. In avoiding presenting the underlying technical aspects of ML to students, educators run the risk of teaching students nothing more than they already know from interacting with ML in other contexts, including those available on technology that students use daily, which are manifold.

Appropriate Selection of Topics

A combination of the other considerations made here, topic selection depends on several factors. First, an ML topic should have a clear path to make it as interactive as possible, particularly interaction with the trained model predictions. It is certainly recommended to allow students to participate in deciding on an application that they find interesting, perhaps constrained by a broad area such as image classification, allowing them to suggest ideas for what domains of data to investigate.

Second, topics should be practical to implement given the time and computing constraints that an educator is subject to; experimentation with different ML domains and data types can give educators insight into which ML applications may be feasible (e.g., smaller models with simple data sets) and which are not (e.g., large sequence models like Transformers, RNNs). Cursory online research will likely be a valuable tool in understanding which classes of problems are suitable for resource-limited classroom ML projects.

Third, perhaps more than many other educational domains, it is important not to bite off more than one can chew with ML due to its inherent complexity and opacity. It is imperative to ground ML lessons in the broad context of how it solves problems and the key distinctions between ML and other computing fields, mainly that it is data-driven at its core. Theory should be introduced as needed and only as needed, and the lessons should otherwise focus on practical realization.

Finally, we feel that a powerful guideline for topic selection is that of the low floor and high ceiling (Resnick & Silverman, 2005). Simply put, models with the potential for compelling predictive capacity and that do not require extensive background knowledge or burdensome implementation are an optimal starting point in searching for appropriate ML topics. We feel that simpler deep learning models like DNNs and CNNs provide a low floor for entry and a high ceiling for performance with the greatest chances of successful delivery and reception to high school students.

Other Considerations

Being that our group lacks expertise in the art and science of education, there are conversations about ML education that we neglected here. Computational topics in K-12 education often deal with notional machines (Tedre et al., 2021), a mental model of how a piece of code is executed in a computer. Machine learning differs greatly from the notional model of classical computer instructions; instead of basic operations organized by looping and branching, ML models (especially DL models) perform a task by performing an enormous chain of uninterpretable operations to transform data into predictions. This new mental model is not incomprehensible, but attention must be paid to the departure from how people understand computers as programmable and sequential rule followers.

Another consideration is that machine learning occupies a strange place in STEM education, particularly from classical computer programming, in that ML systems possess different targets from these other fields. Whereas the outcome of a scientific experiment, mathematical procedure, or rule-based computer program has some sense of objective correctness or lack thereof, ML shifts the quality of system performance from "correctness to effectiveness" (Tedre et al., 2021). The acceptable level of ML system performance is not binary but statistically and ethically determined by its approximate effectiveness. This is because a trained model will never achieve 100% prediction accuracy on new inputs, and in many cases, the definitions of these quality metrics are not so easily characterized. Contending this key difference between ML and other STEM fields complicates ML education, requiring application-dependent discussions about acceptable levels of legitimacy.

Perhaps the most pressing issue in AI by investors, practitioners, educators, and students alike is the ethical issue. Determining the ethical considerations of where to apply AI and ML mitigating societal bias and inequity in the model training itself, is a societal and cultural conversation that is nowhere near settled. An ML model inevitably appropriates the biases present in its training data, an issue whose amelioration requires more input from policymakers and social justice advocates than ML practitioners. We do not address these considerations here, but we must mention that AI and ML ethics are a non-negotiable component of responsible AI education.

Conclusion

After three years of implementing and refining the Machine Learning for High School Teachers workshop we have received positive feedback from secondary educators on our selection of topics and methodologies for introducing machine learning and deep learning into high school curriculum. From the perspective of practical domain expertise, rather than STEM pedagogy, insights from the teacher participants of the workshops can be summarized as follows:

- Teaching modalities which promote active learning are preferable to lecture. This implies project-based implementations, where underlying ML theory is introduced only as necessary. Projects should be data driven, with students participating in collecting and/or analyzing a data set. Interaction with the trained model in a tangible way should be emphasized, making applications involving computer vision or robotics a sensible starting point.
- Simple DNNs and CNNs are reasonable model selections, as they offer high performance potential without being entirely opaque. This motivates students to engage with machine learning material and observe the capabilities of such models and decreases the likelihood of student's becoming frustrated with complex theoretical topics.
- Computational resources and data availability are key ingredients for successfully training a ML model. and if an educator lacks these resources, publicly available datasets and cloud computation services are often suitable for use in secondary education.

Recommendations

Additional surveys which examine the observations, challenges, and missing resources of high school educators integrating machine learning would help focus the attention of STEM education researchers and developers of educational ML tools. Further research which investigates the success of various ML initiatives in secondary education would assist in clarifying which collections of topics and classroom modalities are best suited for fostering engagement and understanding in high school students.

It is important to recognize that the landscape of AI/ML K-12 education evolves very quickly as new toolkits and services are developed, which despite overwhelming educators with an excess of options, opens new applications and possibilities for machine learning education.

Acknowledgements

We acknowledge the financial support from the College of Engineering and Physical Sciences (CEPS), and the Research and Economic Development Division (REDD) of the University of Wyoming to host the workshop.
We also acknowledge the hardware donations from NVIDIA Corp. for use during the workshop, and during the academic year by the workshop participants to introduce ML to high school students.

References

Alpaydin, E. (2020). *Introduction to machine learning* (4th ed., Ser. Adaptive computation and machine learning series). The MIT Press.

Estevez, J., Garate, G., & Graña, M. (2019). Gentle Introduction to Artificial Intelligence for High-School Students Using Scratch. *IEEE Access*, *7*, 179027–179036. doi:10.1109/ACCESS.2019.2956136

Martins, Ramon & Gresse von Wangenheim, Christiane. (2022). Findings on Teaching Machine Learning in High School: A Ten - Year Systematic Literature Review. Informatics in Education. 22. 10.15388/infedu.2023.18.

Pandey, M., Fernandez, M., Gentile, F., Isayev, O., Tropsha, A., Stern, A. C., & Cherkasov, A. (2022). The transformational role of GPU computing and deep learning in Drug Discovery. *Nature Machine Intelligence*, *4*(3), 211–221. https://doi.org/10.1038/s42256-022-00463-x

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

Samuel, A. L. (1959). Some studies in machine learning using the game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229. https://doi.org/10.1147/rd.33.0210

Schmidhuber, J. (2022). Annotated History of Modern AI and Deep Learning. *ArXiv, abs/2212.11279.*

Tedre, M., Toivonen, T., Kahila, J., Vartianen, H., Valtonen, T., Jormanainen, I., & Pears, A. (2021). Teaching Machine Learning in K-12 Classroom: Pedagogical and Technological Trajectories for Artificial Intelligence Education. *IEEE Access*, *9*, 110558–110572. https://doi.org/10.1109/ACCESS.2021.3097962

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (07 2019). Envisioning AI for K-12: What Should Every Child Know about AI? *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 9795–9799. doi:10.1609/aaai.v33i01.33019795

Yu, Yanfang & Chen, Yuan. (2018). Design and Development of High School Artificial Intelligence Textbook Based on Computational Thinking. *OALib*. 05. 1-15. 10.4236/oalib.1104898.

Author Information

Fletcher Wadsworth
https://orcid.org/0009-0002-5383-3498
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Josh Blaney
https://orcid.org/0009-0002-2713-3156
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Matthew Springsteen
https://orcid.org/0009-0001-0543-1243
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Bruce Coburn
https://orcid.org/0009-0002-4916-5095
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Nischal Khanal
https://orcid.org/0009-0007-4969-7637
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Tessa Rodgers
https://orcid.org/0009-0003-5356-5486
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Chase Livingston
https://orcid.org/0009-0001-4001-1607
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA

Suresh Muknhahallipatna
https://orcid.org/0000-0003-4153-524X
University of Wyoming
1000 E University Ave., Laramie WY, 82072
USA
Contact e-mail: *sureshm@uwyo.edu*

Appendix

A – Online ML activities

- *Quick, Draw!* is a web-based ML activity where users are prompted to draw different objects in 20 seconds. After drawing the objects, the online ML model will attempt to classify the sketches correctly. After classification, users can see example drawings of the prompts used to train the model and how the model assessed the incorrectly classified drawings. Quick, Draw! was developed by Google Creative Lab and is available at https://quickdraw.withgoogle.com/.
- *Scroobly* is a web-based ML activity requiring a webcam. Users are prompted to draw a humanoid doodle, and the ML models map the user's live motion, captured by the webcam, onto the doodle. Scroobly was developed by Google Partner Innovation and bit.studio and is available at https://www.scroobly.com/.
- *Thing Translator* is an online ML activity requiring a camera. Users are prompted to take pictures of objects in their environment, and the ML model identifies and translates the object's name to a language specified by the user. Dan Motzenbecker and Google Creative Labs developed Thing Translator, available at https://experiments.withgoogle.com/thing-translator.
- *Lobe AI* is an application that allows users to create and train models without programming. Users can take or upload images, label images by class, and train a classifier model using a streamlined graphical user interface. Microsoft developed lobe AI, which is available for download at https://www.lobe.ai/.

B – Drones and Robotic Platforms

- The DJI Tello EDU is a quadcopter drone designed for programming and robotics education. It has an inbuilt camera with 720P HD video transmission. The Tello may be interfaced with a mobile application or programmed and controlled using Python, Scratch, or Swift.
- The TurboPi is a wheeled robot car kit with a Raspberry Pi as the platform controller. TurboPi has omnidirectional mecanum wheels for simplified control, as well as a camera, ultrasonic distance sensor, and color detector comprising its interfaceable sensors. TurboPi is developed by Hiwonder.
- The PuppyPi is a quadruped robot controlled with a Raspberry Pi using the Robotic Operating System (ROS) software platform. Navigation is actuated with 8 coreless servo motors, and a camera and 2-D LiDAR sensor provide the main sensor fusion. PuppyPi is developed by Hiwonder.
- The JetHexa is a hexapod robotic platform controlled with a NVIDIA Jetson Nano using the ROS software platform. Similar actuators and sensors as on the PuppyPi are available on the JetHexa. JetHexa is developed by Hiwonder.