Reorienting Machine Learning Education Towards Tinkerers and ML-Engaged Citizens

by

Natalie Lao

S.B., Massachusetts Institute of Technology (2016)
M.Eng., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author ....
Department of Electrical Engineering and Computer Science
August 28, 2020

Certified by..
Harold (Hal) Abelson
Class of 1922 Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Reorienting Machine Learning Education Towards Tinkerers and ML-Engaged Citizens

by

Natalie Lao

Submitted to the Department of Electrical Engineering and Computer Science
on August 28, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Artificial intelligence (AI) and machine learning (ML) technologies are appearing in everyday contexts, from recommendation systems for shopping websites to self-driving cars. As researchers and engineers develop and apply these technologies to make consequential decisions in everyone's lives, it is crucial that the public understands the AI-powered world, can discuss AI policies, and become empowered to engage in shaping these technologies' futures. Historically, ML application development has been accessible only to those with strong computational backgrounds working or conducting research in highly technical fields. As ML models become more powerful and computing hardware becomes faster and cheaper, it is now technologically possible for anyone with a laptop to learn about and tinker with ML. This includes audiences without programming or advanced mathematics knowledge, who can use public ML tools to construct both useful technical artifacts and meaningful socio-ethical artifacts around ML. My research investigates the following question: How do we reorient ML engineering education to transform "ML consumers" to be "ML contributors"? In order to illuminate this problem, I have created a Machine Learning Education Framework. In this dissertation, I present the framework and three applications of it: (1) a course based on the framework that aims to develop ML self-efficacy in general college-level audiences; (2) a curriculum rubric based on the ML Education Framework and analyses of 6 ML/AI courses using the rubric; and (3) two public ML tools for high school and above ML tinkerers that empower them to create personal ML applications through image and audio classification.

The ML Education Framework is organized into three interconnected categories: Knowledge (General ML Knowledge, Knowledge of ML Methods, Bias in ML Systems, Societal Implications of ML), Skills (ML Problem Scoping, ML Project Planning, Creating ML Artifacts, Analysis of ML Design Intentions & Results, ML Advocacy, Independent Out-of-Class Learning), and Attitudes (Interest, Identity & Community, Self-Efficacy, Persistence). The three top level categories of the framework were chosen to encompass the holistic process of teaching people ML and ensuring continued participation in the field. "Knowledge" about ML will be necessary for individuals to thrive in today's ML-driven world and builds foundations for further engagement with the technology. ML "Skills" define specific actions and activities that students need to do and practice in order to become ML contributors. The category "Attitudes" reflects long-term societal goals for educational initiatives in ML, and addresses the perspectives that students should develop through an ML course. This framework can be used in creating new ML curricula, in analyzing the results of ML interventions, and in situating support tools within ML education for students of a variety of backgrounds and levels.

Today a large portion of the world solely resides in the "ML consumer" population. Regulation of ML technologies is nascent. As policymakers think about reasonable legal structures, the public should be able to actively participate in these discussions. It is imperative that we empower these "ML consumers" to become "ML tinkerers" and "ML-engaged citizens".

Thesis Supervisor: Harold (Hal) Abelson
Title: Class of 1922 Professor of Electrical Engineering and Computer Science

Acknowledgments

I have received a lot of help and support throughout the past 3 years of my PhD at MIT. First, I want to thank my thesis committee: Hal Abelson, Anantha Chandrakasan, and Eric Klopfer. Thank you for your mentorship and support throughout this whole process. Hal, I've taken your classes and worked on research with you since my junior year of undergrad at MIT. Seeing your pioneering approach over the past years has really inspired me to work in academia and do research related to democratizing technology and education. Anantha, I've also known you since my undergrad years at MIT. You've always pushed me to think more deeply when it comes to research, and even though you're incredibly busy as dean, your thoughtful feedback has helped me consider how to maximize the real-world impact of my work. Eric, your expertise in education has been invaluable to the quality and rigor of this work. Your advice has really helped me refine my Machine Learning Education Framework and the corresponding rubrics into something that hopefully can be used by more people.

I also want to thank the people who were instrumental in the papers I've written during my PhD. Irene Lee, the 6.S198 staff, and my 6.S198 students helped make the Deep Learning Practicum course a wonderful experience. The Google PAIR and TensorFlow.JS teams were also extremely supportive during this process: I recall we were using deeplearn.js when it first came out, so sometimes we would run into bugs while making the course materials. The folks at Google would help us fix these issues within a few days. I would also like to acknowledge the staff and students from the Internet Policy Research Initiative and the MIT App Inventor team for helping create PIC and PAC, and running our high school workshops. Special shout-outs to Danny Tang, Yuria Utsumi, Nikhil Bhatia, Evan Patton, Jeff Schiller, Karen Lang, Selim Tezel, and Marisol Diaz. Also a big thank you to Ingrid Roche at the Boston Latin Academy for letting us run workshops in her classes.

I had many meaningful and rewarding conversations with people in the ML and AI education community when creating my framework and doing my evaluations, and these conversations were instrumental to my research process. Thank you for taking the time to speak with me: Richard Jung and Nicholas Luu from Technovation, Sandhya Simhan and Ortal Arel from deeplearning.ai, Blakeley Payne and Randi Williams from the MIT Media Lab, and Teemu Roos from the University of Helsinki.

Finally, I would like to thank my friends, my parents, my fellow graduate students, my fiancé Nicholas Kwok, and my dog Sophie for all of their support throughout these years. MIT and learning how to do research would have been less fun without you all.

Contents

List of Figures

List of Tables

1

Introduction

This dissertation focuses on democratizing machine learning (ML) for the general public through reorienting machine learning education to transform ML consumers to be ML contributors. I present an education framework for ML that will help inform future decisions and trajectories for general ML education, ML engineering, and ML-related HCI work (see Figure 1-1).

In the rest of this chapter, I will elaborate on the vision of this work and position this work in the larger history of artificial intelligence, machine learning, and democratization of technology.

In Chapter 2, I present relevant literature from the education research field that are heavily referenced in this work, including self-efficacy, computational identity, and computational thinking.

In Chapter 3 I introduce my framework for ML education, and also clarify the knowledge and skills that are most essential for the general public. I present a logic model of this framework with linkages.

In Chapter 4, I present the curriculum, research results, and pedagogical findings for an actionable deep learning course I designed and taught for 2 semesters at the Massachusetts Institute of Technology based on my ML Education Framework.

In Chapter 5, I present a rubric for analyzing ML curricula created from the principles of the ML education framework on a 0-4 scale. I apply the Curriculum Rubric in analyzing 6 AI and ML courses with input from the original course developers,

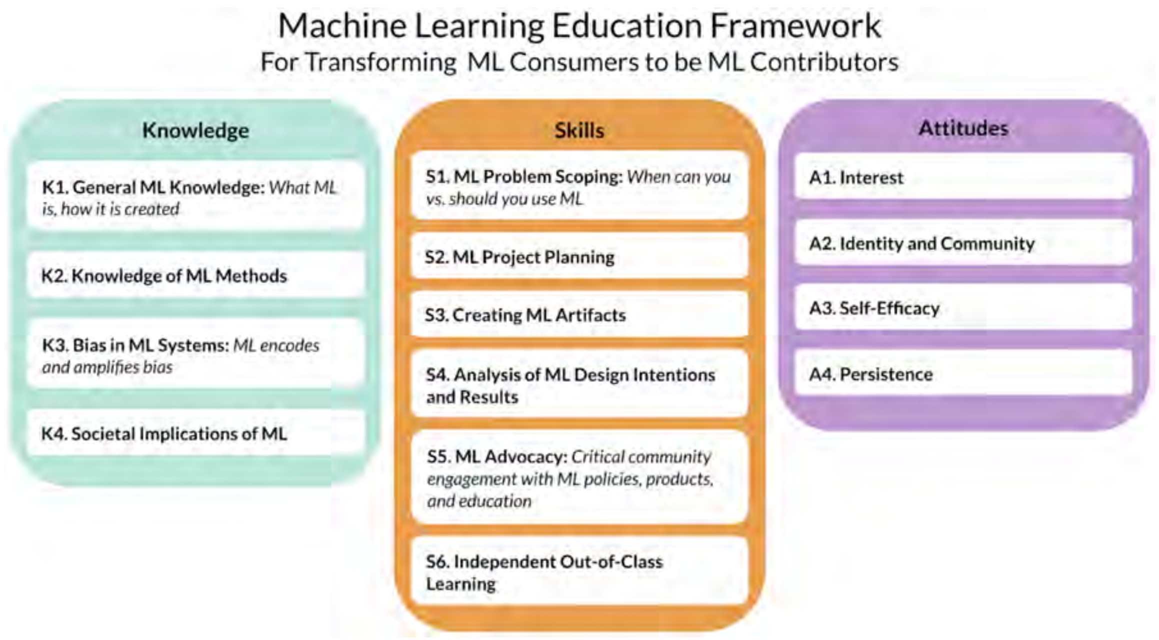

Machine Learning Education Framework
For Transforming ML Consumers to be ML Contributors
Knowledge
K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
K3. Bias in ML Systems: ML encodes and amplifies bias
K4. Societal Implications of ML
Skills
S1. ML Problem Scoping: When can you vs. should you use ML
S2. ML Project Planning
S3. Creating ML Artifacts
S4. Analysis of ML Design Intentions and Results
S5. ML Advocacy: Critical community engagement with ML policies, products, and education
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence

Figure 1-1: The Machine Learning Education Framework for Transforming ML Consumers to be ML Contributors.

including the course I presented in Chapter 4. My analysis using the Curriculum Rubric reflects the different motivations of K-12 AI courses, technical ML courses, and AI courses for general adult audiences: K-12 courses tend to focus on analysis of ML products and systems in relation to society, technical courses dive deeply into math and hands-on coding experiences without much discussion of analysis or societal implications, and general adult courses tend to aim for fostering interest and encouraging students to take technical courses. I offer insights into designing and implementing ML curricula based on these reviews.

In Chapter 6, I describe PIC (Personal Image Classifier), a public ML tool that I created for high school students to train image classification models and apply them in mobile applications built using MIT App Inventor. I discuss the HCI and engineering considerations of creating PIC and how the ML Education Framework informed the design and implementation of this tool.

In Chapter 7, I describe PAC (Personal Audio Classifier), a public ML tool similar to PIC that I created for high school students to train audio classification models and apply them in mobile applications built using MIT App Inventor. I discuss the technical challenges of PAC that differed from PIC and the improvements made to the user interface.

Finally, in Chapter 8, I propose new research questions relating to the field of machine learning education and review the contributions of this dissertation.

1.1 Vision: Everyone can understand and tinker with machine learning

Artificial intelligence (AI) is “the study of agents that receive percepts from the environment and perform actions” [110]. It includes logic systems crafted from general *deductive* principles that solve specific problems, as well as machine learning (ML) models trained on large amounts of data to *inductively* find patterns. Over the past decade, machine learning has been a driving force in economic growth internationally and has had a revolutionary impact on the world stage. Large corporations and governments are pledging billions of dollars to ML research and development. In 2017, China laid out a plan to lead the world in artificial intelligence by 2030 [85], and in 2018, the White House directed agencies to prioritize investigating this “emerging technology” [95]. The rapid growth of machine learning necessitates a push for mass ML education so that there can be broader participation in the ML innovative and regulatory processes.

Currently, ML development is still widely considered to be accessible only to high-level technical researchers at universities or large companies. A 2017 study by The Royal Society interviewed members of the public in the UK, finding that a majority of participants knew “little to nothing” about machine learning [84]. While many of the participants were aware of technologies that use ML, very few were aware of how the technology worked, “even at a broad conceptual level”. Another study found that even amongst UX designers working on projects that involved ML, a lack of understanding was common. One participant referred to ML as “black magic”, stating that “designers don’t understand the constraints of the technology and how to employ it appropriately" [26]. The global push to advance the field will inevitably lead to efforts in educating the larger public so that more people are able to contribute to ML innovation and policy-making. So, how should we be teaching this topic to a broader and less technical audience? What aspects of the technology do they need to know in order to make useful contributions? Can these considerations perhaps bring to light pieces that may be missing from the current modus operandi for ML engineering education?

Technology leaders such as Massachusetts Institute of Technology have stated that "machine learning is the new literacy" [94], pushing for mass general education on ML. As we move towards this goal, it will be important to consider how ML differs from classic algorithms and how we can tailor educational frameworks and pedagogical practices based on these differences. In order for an ML system to "work", it is dependent on the availability of high-quality data, scientific insights on features, appropriate model architectures, and computational processing power. Instead of making generally deterministic programs, ML developers create systems that automatically and stochastically learn behaviors from large amounts of data. This results in powerful statistical algorithms that can be hard to debug and understand on a detailed level, such as when trying to find the cause of a single error in a large neural network [114]. Due to its stochastic nature, machine learning differs from classic computer science and even other artificial intelligence topics. Thus, if we want to educate critical thinkers who are able to apply ML knowledge, tools, and techniques to a variety of problems, we should consider which concepts and skills are specifically necessary for machine learning education.

The current standards for ML education are college or professional level courses that target people who have a background in computer science or mathematics, with the goal of giving them the ability to become ML engineers or researchers. These courses cover the technical aspects of ML algorithms and methods, amassing mathematical building blocks to help students develop a deep technical mastery over the inner workings of ML models. This forms a rather high barrier to entry for a field that society is attempting to ubiquitize. Recently, educators have started to create courses aimed at the general public and even young children and there is a need to distill the key abstractions that can help teach ML in useful, actionable ways. As we build upon these more inclusive educational initiatives, we should think carefully about our educational goals for general audiences.

What should they know?

What should they be able to do?

Why and for what purpose?

I will discuss three key reasons for why it is important to teach ML to everyone: (1) to help people navigate the world they live in, (2) to empower them to participate in discussions regarding policies around ML, and (3) to prepare for a future of work in a world that is tightly integrated with AI and ML.

First, people need to understand the world they live in. ML is poised to influence every facet of our lives and work. Currently, nearly everyone with internet access is consuming an ML product or some output of an ML system in their daily lives, whether they know or recognize these ML products or not. In the next decade, the prevalence of ML is expected to increase. As people consume these products, they need to be able to understand enough about them to use them effectively and safely. For example, when people are on social media platforms like Facebook or YouTube, they usually see that the content they are recommended are related to things that they have liked or interacted with in the past [18]. While this can be useful, it can also cause people to be enclosed in "filter bubbles" or "echo chambers" where they aren't aware of what important information they are missing [141]. If consumers of social media don't actively try to curate and amend what they see, it's easy for them to never interact with different or opposing ideas and develop a distorted perception of reality. Consumers also need to be able to identify and critically interpret ML outputs to navigate the world. Let's consider the issue of photorealism. Most people are aware of Photoshop and other photo manipulation tools and know to have a healthy mistrust of photorealistic images that don't make sense. However, recent advances in ML have led to the ability to generate deepfakes that realistically replicate another human's voice and expressions in video and audio [91]. While this application of ML could revolutionize the entertainment industry, it could also lead to chaos as people who are unaware of deepfakes misplace their trust in artificial videos created for political sabotage. Ideally, we would want to educate people such that they know such artifacts exist and provide them with the knowledge and tools to critically consider what they see.

The second key reason that we need to educate the general public in ML is because people need to be able to participate in discussions regarding policies around ML technologies. People can't give thoughtful opinions unless they know something about the technology, its potential benefits and consequences, and have some way of critically thinking about these factors in relation to their own lives and in creating a future society that they would be happy to live in. In governance, for example, many companies are developing algorithms to assign risk scores to arrestees, which are currently used by U.S. Courts to determine whether or not people are low risk enough to be released on bail pending trial [96]. These tools have been very controversial due to researchers pointing out risks with racial bias [5]. While these algorithms have potential for improving slow governmental operations, people want to have a say in what kinds of algorithms are used, how they are used for governance, and what the processes for auditing or improving on them are. Another big area of debate surrounding ML has to do with privacy. Smart speakers and virtual assistants have become very popular products, and companies like Amazon have revealed that Alexa voice recordings are sometimes sent to contractors to be transcribed for research and product improvement purposes [21]. In Alexa's case, people were concerned that these recordings could potentially be traced back to individuals. This type of data sharing is on by default in most such devices, and there has been a lot of debate around protecting consumer privacy interests, especially for people who may not even know that this is happening or understand the potential implications. Regulations and legal precedents around ML technologies are just starting to be formed for both governmental usage of ML and corporate applications of ML, so now is a great time for the people who will be affected to contribute and make their voices heard in policy-making.

Finally, it's highly likely that the future of work will be in a world that's tightly integrated with AI and ML. The world will need more people who are trained to work with ML (e.g. judges who need to interpret outputs from ML algorithms), and people who are able to develop and build these types of systems. Providing widely accessible general ML education will help this societal job transition process along.

As ML products and services become more ingrained in everyone's daily lives, people need to be able to understand how they work to use them effectively and safely, and people need to be able to participate in discussions around policies and regulations. This leads us to want to educate general audiences such that they have the knowledge necessary to be critical consumers and ML-engaged citizens. There will also be a big societal push towards training more and more people who can build ML systems and work with ML. This motivates teaching general audiences the basics of creating and tinkering with ML systems.

1.2 Photography: An Analogy for Democratizing Machine Learning

Many technologies have undergone the democratization process throughout history. In the still-ongoing makerspace movement, individuals are empowered to seize the power of personal fabrication. Tools such as laser cutters, CNC machines, and 3D printers allow people to create custom physical artifacts without access to expensive manufacturing resources. During the Computer-Aided Design movement, CAD software was reimagined by Negroponte and Friedman as mechanical partners that could work with humans. CAD software was later retooled in a way that allowed it to help dwellers participate in the design of buildings, an area that had previously been the territory of elite architectural experts [76]. Even in programming, tools like MIT App Inventor have been designed so that youth can be empowered to create personally meaningful mobile applications [102]. As we tackle the big question of how to democratize machine learning and make it more accessible to a wider audience, there is another technology whose history and development may be an interesting analogy to examine: photography.

In the 1800s, William Talbot, a scientist who researched spectrochemical analysis, established the first negative/positive photographic process [97]. This spawned the first major stage in the history of photography, when it was mainly a “science”. The key research question was “How do I take a clear picture?” and the pursuit of this “science” required high technical knowledge of chemical processes, high time commitment, and high monetary resources. In this first stage, photography was more of an interesting novelty than anything commercially useful, and was available to only a small group of technical experts and wealthy individuals.

As the technology became further refined, it became slightly faster and cheaper to take a photograph. This pushed photography into the second stage of its democratization, as it became more of an “engineering discipline”. Engineers attempted to improve upon and perfect the technical methodologies behind photography during this time, leading to the creation of the first cameras, which allowed more individuals (though still mostly of the high wealth and high technical expertise bracket) to explore the technology.

The third phase of photography started in the early to mid 1900s, when the advent of easy-to-operate portable cameras allowed for more everyday use by the press [113]. Photography became viewed as more of a “craft” than a technical discipline during this stage. The key questions surrounding photography also shifted from technological research questions to more subjective and abstract ones such as “How do I take an interesting picture?” Around this time, privacy and copyright laws surrounding photography started to become defined.

Finally, we are currently in the fourth stage of photography’s democratization, when it has become almost a “natural activity”. Nearly everyone carries a high-tech camera with them at all times via their smartphones and mobile devices. The user interfaces of these cameras are thoughtfully designed to be very efficient and easy to operate for a wide audience, requiring minimal technical knowledge. Modern cameras are both extremely fast and relatively inexpensive, making them easily accessible. Additionally, members of the public are now able to contribute to photography-related policy discussions surrounding privacy and intellectual property ownership, and ask questions like "What should or shouldn't we be using this technology for?" Recently, smartphone cameras have even been used as a tool for activism — they are able to almost instantly give anyone a voice and a media platform in documenting injustice [3]. In this fourth stage, we consider photography to be a technology that has been successfully democratized.

Machine learning is currently in the second stage of the democratization analogy a la photography, where it is considered an "engineering discipline". Developers no longer spend most of their time fiddling with hardware; and many don't even need to code algorithms from scratch, as there are a plethora of ML libraries and packages available. Engineers are now working faster and at higher levels of abstraction than before in applying these technical methods to real world problems. If we want to help machine learning arrive at the next stages of democratization — machine learning as a "craft" and as a "natural activity" — it is important to consider three areas of improvement:

- **Engineering**: Just as the decrease in cost and increase in speed of portable cameras hailed the advent of photography as a "craft", decreasing the cost and increasing the speed of creating and running powerful models will do the same for machine learning. This will involve innovative technical solutions by both hardware specialists and model theorists. Computing hardware for machine learning needs to be made more powerful and more affordable, and ML models need to be more compact without compromising on function. There has been a lot of ongoing research in these fields, with some successes already. As electrical engineers and material scientists create cheaper and faster chips for ML, ML developers are creating highly functional models that are small enough to run on laptops or mobile phones [48, 52, 36], and entire ML libraries for training and running ML models on the browser [116].

- **Human-computer interaction (HCI)**: HCI research will be essential in making machine learning interfaces usable, understandable, and interpretable. As we incorporate machine learning models into more workflows and processes across industries, we need to study how these ML systems interact with human decision-makers. Creators of these systems need to carefully and thoughtfully design them to decrease potential risks and increase human-machine synergy. Additionally, HCI experts will be crucial as we develop more public ML tools that allow new populations of people to create, train, debug, and analyze machine learning models.

- **Education**: Expert educators and educational researchers will need to define learning objectives and pedagogical practices for teaching machine learning to students of various grade bands and backgrounds. This will involve designing curricula and identifying appropriate machine learning tools to use in classroom settings. Much of this work is expected to be in close collaboration with machine learning experts.

Since our intended audience of the general public is quite broad, it will be useful to circle back to the guiding questions presented in Section 1.1:

1. What should they know?

2. What should they be able to do?

3. Why and for what purpose?

Let's start with question 3. As discussed previously, machine learning is rapidly being integrated into every aspect of our society and individuals' daily lives. It is very difficult to predict how machine learning will be utilized when it reaches the "natural activity" stage of democratization, but it will soon become imperative that everyone feel empowered living with machine learning. Currently there are two populations: the creators of ML technologies, which mostly consists of engineers and researchers, and the consumers of ML technologies, who know little about the workings or limitations of the products they are using. There are already serious consequences from a lack of ML literacy: in this context, photo manipulation software is a good analogy for deepfake technology. Another issue tangential to ML is the mining of big data to train models. People who don't understand how their personal data can be collected, sold, and used to encroach on their privacy may unwittingly divulge personal information that later harms them. There are many more examples of potential harms when an ML-illiterate population resides in an increasingly ML-dominated society, and it is important that we transition the current consumers of ML to ML tinkerers, critical consumers of ML, and ML-engaged citizens.

From these motivations, general ML education involves two types of empowerment: first, empowering people so that they feel "I am able to do something with the technology"; and second, empowering them so that they feel "I am able to do something about the technology". These two mental constructs lead to question 2. ML critical consumers will need to be able to participate in discussions about ML regulations and products, and ML tinkerers may want to use the technology to create or modify personalized ML artifacts instead of solely consuming off-the-shelf products.

With regard to question 1, the ML tinkerer/critical consumer populations need to know enough about ML and its potential for good and bad in order to do the tasks addressed above.

1.3 A Brief History of AI

It is difficult to predict what the world will look like if machine learning achieves the fourth stage of democratization (ML as a "natural activity"). Perhaps we will see personal systems that are constantly collecting all sorts of data around us and about us. Maybe these systems will be able to automatically configure themselves on the fly to complete a wide range of tasks that a user can ask them to do and clearly explain their own processes and actions. As we work towards harmoniously incorporating ML technologies within our societies and our world, let's first consider a brief history of AI so that we can recognize how the way AI is approached today (since 2010) differs from how it was approached in the past.

The field of artificial intelligence grew from the millenia-old human pursuit of trying to understand our own intelligence. In 1950, Marvin Minsky was one of the first people on record to attempt to artificially recreate intelligent processes of the human mind. He built the Stochastic Neural Analog Reinforcement Calculator (Snarc), which consisted of 40 models of neurons constructed from motors, clutches, and capacitors; and held electrical charge for a short period of time, simulating memory [82]. While not practically useful, Snarc is often credited as the first neural network simulator. Seven years later, Frank Rosenblatt conceived of the perceptron, a "machine" that approximated a "hypothetical nervous system" by updating weights in a linear classification algorithm when given new data, simulating learning [108]. Shortly after that in 1958, Arthur Samuel first coined the term "machine learning" through a checkers-playing computer program that employed minimax search and alpha-beta pruning, which was able to play against humans at an amateur level [111]. Following these inventions, wildly optimistic predictions about solving the nature of consciousness and creating an artificial being of human-level intelligence led to an outpouring of interest and funding from the military and government. However, researchers soon hit a high wall when applying these inceptive, biologically-inspired models to real world problems — the perceptron had very little success when utilized in image classification tasks since it could not handle nonlinearities, and limited computational capacity at the time inhibited research into assembling large neural networks out of mathematical/electro-mechanical models of neurons. This led to the first AI winter, which involved a dearth of research funding for AI.

Public interest began to pick up again in 1977, when IBM's Deep Blue computer employed minimax and alpha-beta to play chess at the highest levels [49]. Deep Blue was implemented on a machine that was much more computationally powerful than the one used by Arthur Samuel in 1958, and it was able to successfully defeat Garry Kasparov, a human chess world champion. This led to another surge of funding for AI by commercial industries and government, although the research trajectories this time pointed towards building expert, rule-based systems founded upon chains of logic instead of systems that simulate biological processes. However, researchers once again fell short of expectations of creating tools that could actually be used in the real world, as these expert systems made of hand-coded rules could not deal with uncertainty and were unable to extend beyond their predefined specialized domains. This cycle of overhype and disappointment led to a second AI winter.

During this time, Rumelhart, Hinton, and Williams reintroduced the concept of backpropagation by applying it to deep feedforward neural networks (multilayer chains of perceptrons whose outputs are fed as inputs into other perceptrons) [109]. Their experiments demonstrated backpropagation's ability to allow the hidden units of neural networks (the units between the input and output layers) to encode useful internal representations. This work slowly popularized the use of backpropagation in deep neural networks, which has had a huge influence on defining the goals and limits of modern machine learning.

The third wave of AI that we have just started experiencing over the past decade can be characterized by computational power and extreme practicality. Compared to the previous two waves, there is less interest in artificial general intelligence (AGI), which deals with the traditional goals of understanding human intelligence and creating an artificial consciousness comparable to humans. Instead, most of the funding (and thus, most of the work) has been directed towards creating useful solutions to well-defined and narrow real world problems that have clear economic value. Machine learning techniques and deep neural networks in particular have once again taken center stage. Although much of the mathematics behind these methods had already been proposed decades ago, the powerful computing hardware we have access to today is able to compute in weeks what would have previously taken decades. Unlike what led to the previous AI winters, ML researchers and engineers have already made real headway in applying narrow ML models to increase efficiency in businesses and generate entirely new sectors of commerce. Due to the economic value already being produced by ML, the current wave of interest in ML is unlikely to undergo another big freeze, and we can expect to see ML technologies seep into more products and services relevant to our daily lives.

1.4 Defining Artificial Intelligence, Machine Learning, and Deep Learning

Now that we have some background on the history of this field, let's define some key terms that will be used in the rest of this work. While the term "artificial intelligence" has taken on various meanings over the past several decades, becoming what many refer to as a "suitcase word", today it most often refers to the broad field of applying models to real world data. AI includes logic systems crafted from general deductive principles to solve specific problems (e.g. human-coded decision trees, alpha-beta pruning and minimax), as well as machine learning, which refers to models trained on large amounts of data to inductively find patterns.

Machine learning includes algorithms such as k-nearest neighbors, naive Bayes, and random forests, as well as neural networks. Unlike other AI topics or classic algorithms, ML involves statistical models derived from a combination of human written components and behaviors learned from large data sets, which become difficult to interpret and debug, especially when the models balloon in size [114].

Deep learning is a subtopic of machine learning involving neural networks with at least one hidden layer (defined as any additional layer in between the input and output layers), and is often very data-hungry, thus requiring powerful computational resources. Deep learning models/deep neural networks have proven to be effective by themselves when applied to a wide range of tasks, from image classification to predicting stocks, but ensemble methods combining multiple ML techniques into one system are becoming more popular.

Now that we have summarized the stages of technology democratization and the history of AI, let us consider how to expand the use and understanding of AI for the general population. In the remainder of this thesis, I will discuss how educational efforts can best support these goals.

2

Background: ML Education and Educational Theories as Applied to Teaching ML

In this chapter, I summarize the types of ML/AI courses and learning resources that are currently available for three types of audiences: technical, general adult, and K-12. I then discuss the theoretical considerations from education that influenced my work in designing the ML Education Framework and teaching ML courses, specifically that top philosophies in education say that the best way to teach students is to empower them to create personally meaningful and useful ML artifacts. I will focus on four main ideas in education theory that are highly applicable in thinking about teaching complex, technical subjects to general audiences: self-efficacy, constructionism, computational action, and experiential learning.

2.1 The Current State of Machine Learning Education

There are three main audiences that have emerged for ML education: technical audiences such as engineers and researchers, general adult audiences who are starting to interact with ML at work, and K-12 audiences. Technical courses are very well established in terms of content and style, and are widely accessible, but have a high barrier of entry to understanding the material and being able to complete the course. They generally begin from a math-first approach before moving on to coding ML models. General adult and K-12 courses are not as well established and have only begun to take shape over the past 2-3 years. There are limited options for courses in general adult education, and while there are more available educational resources in K-12, most of them are a mix of short, one-shot activities and workshops with few full courses. These last two categories are the type of education initiatives we need for our original goal of transforming ML consumers to be ML contributors.

2.1.1 Technical ML Courses

Over the past few decades, most ML courses have been targeted towards highly technical college and professional level computer science students. These courses generally start by teaching the detailed mathematics of ML during lectures (such as the perceptron algorithm or linear regression) and then asking students to apply the theories they learned to making proofs and answering questions in problem sets. These problem sets often involve translating mathematical concepts into code and running programs on remote computing clusters.

Full-semester undergraduate or graduate courses on ML are available to enrolled students at many educational institutions (e.g., introductory ML courses at Stanford University [27] and New York University [83]). Many of these courses are even very easily accessible for free online (e.g., MIT Open Courseware [115] or Coursera [89]). These courses are typically intended for computer science (CS) students with an existing background in the material, and aim to provide a significant level of technical depth as well as a wide breadth of ML topics. Shorter, primarily online courses or materials are also offered by some companies (e.g., Coursera [90], fast.ai [32], or Google [24]). These courses may be more targeted towards software engineers focused on practical implementation or towards technical product managers working with ML teams, and usually cover less depth than university courses.

Some recent efforts, such as Embedded EthiCS @ Harvard [132] or the Responsible Computer Science Challenge [34], have produced materials focused on AI ethics that are intended to integrate into undergraduate CS courses. Of course, such courses often have high barriers to entry in terms of math and computer science knowledge. Research has shown that these types of courses can be inaccessible for students who do not have a strong foundation in CS, probability, calculus, or linear algebra. When researchers interviewed students on their perceptions of learning ML, math was consistently a source of anxiety and a barrier to learning [119].

2.1.2 General Adult ML Courses

Over the past few years, there has been a push to make AI education more accessible to the public. Elements of AI is a large-scale program that has gained renown for having international impact in increasing AI and ML literacy. Elements of AI is a Finnish course created by the University of Helsinki and Reaktor in 2018 as a public resource to help citizens "cope with society's ever-increasing digitisation and the possibilities AI offers in the jobs market" [122]. In 2019, the Finnish government sponsored the course to be translated and opened to the world with the goal of giving a practical understanding of AI to the entire population of the European Union. It has now been completed by over 430,000 people from 170 countries [92].

Some companies and organizations are also starting to offer courses for adult audiences. Many of these courses target people who are working with or around ML technologies but are not necessarily the ones doing the development (for example, product managers, UI/UX designers, or people working on business strategies) [90]. There are also other courses being developed that target a more general adult audience with the purpose of helping adults better understand how AI will affect their lives in the future. These courses for general adult audiences are typically much shorter than technical courses and don't go into as much depth. There are also very few of these courses targeted towards general adult audiences compared to technical courses. Most courses of this type came out within the past 2-3 years. The standards for what content should be taught and how it should be taught are also not as well developed.

2.1.3 K-12 ML Courses

The third area of ML and AI education is in the K-12 community. As everyone looks to the future of work, there's been a push to get ML and AI curricula into primary and secondary schools. Educators and researchers have been creating collections of smaller, one-shot activities and workshops for AI and ML that can be integrated into other classes. Over the last 2 years, the community has built several blocks-based programming tools and data-oriented activities that abstract away a lot of the model architecture and tuning steps associated with building machine learning models from scratch (e.g. Cognimates [28], MachineLearningforKids [68]). The MIT AI Education Initiative has also compiled a collection of tools and workshop curricula [4].

Some full courses for K-12 audiences are also starting to be created. These courses are not usually meant to be integrated within formal schooling, but are more for extracurricular or at-home education. The Technovation AI Family Challenge aims to teach kids a breadth of AI through using a collection of public ML tools mentioned previously. Technovation is an international organization that runs community-based learning programs with the goal of teaching girls and families about tech innovation and entrepreneurship [16]. In 2019, they launched the AI Family Challenge [124], an AI education challenge for families of children 16 and under. The program offers a free, hands-on AI curricula with ML units, as well as an international mentorship forum, and has drawn over 2,800 families from over a dozen countries, including Bolivia, Uzbekistan, Spain, and the United States [123]. AI-in-a-Box is another long curriculum that centers around programming a robot using a blocks-based language [53].

AI4K12 Initiative

In addition to private corporations and educational institutions pushing the democratization of AI, governments are also trying to set nationwide educational standards, often motivated by considering the future of work. In 2017, China released guidelines for becoming a global innovation center in Artificial Intelligence by 2030 [112]. The

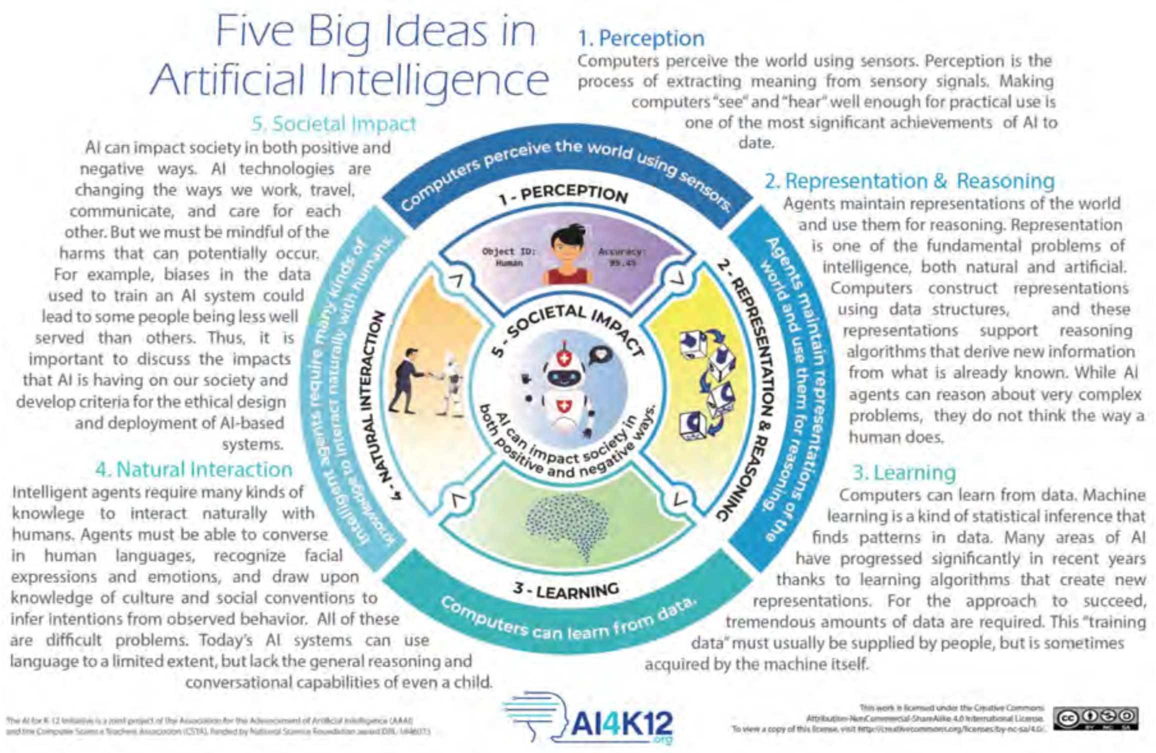
United Kingdom began creating an AI policy for formal education & training, research and development, partnerships, infrastructure, and regulation in 2018 [17]. In the United States, the Computer Science Teachers Association (CSTA) and the Association for the Advancement of Artificial Intelligence (AAAI) sponsored the AI for K-12 initiative (AI4K12) in 2018. The initiative brings together educators of all grade bands, technology intervention specialists, and leaders from academia and industry to create "(1) national guidelines for AI education for K-12, (2) an online, curated resource Directory to facilitate AI instruction, and (3) a community of resource and tool developers focused on the AI for K-12 audience" [130].

In 2019, this initiative presented an initial list of "Five Big Ideas in AI" along with some suggestions of what students should be able to do at each grade band in relation to these topics [37]. As shown in Figure 2-1, the five ideas involve understanding: (1) the role of sensors in helping AI systems perceive the world, (2) how representations of information as data structures support reasoning algorithms, (3) that there are machine learning algorithms that can improve themselves when provided with large amounts of training data, (4) the need for AI agents to support natural interactions with humans, and (5) both positive and negative implications of AI in society.

As of July 2020, AI4K12 has also presented a detailed list of topics that students should know and relevant tasks students should be able to do for Big Idea 1 (Perception) across K-12 grade bands [131]. These items cover 3 concept areas, which are further broken down into specific topics:

- Sensing: Living Things, Computer Sensors, Digital Encoding
- Processing: Sensing vs. Perception, Feature Extraction, Abstraction Pipeline: Language, Abstraction Pipeline: Vision
- Domain Knowledge: Types of Domain Knowledge, Inclusivity

The Big Idea #1 knowledge items and tasks are detailed in Table 2.1 and involve using AI systems, explaining how AI works, and simulating some AI algorithms for select applications.


Five Big Ideas in Artificial Intelligence

1. Perception

Computers perceive the world using sensors. Perception is the process of extracting meaning from sensory signals. Making computers "see" and "hear" well enough for practical use is one of the most significant achievements of AI to date.

2. Representation & Reasoning

Agents maintain representations of the world and use them for reasoning. Representation is one of the fundamental problems of intelligence, both natural and artificial. Computers construct representations using data structures, and these representations support reasoning algorithms that derive new information from what is already known. While AI agents can reason about very complex problems, they do not think the way a human does.

3. Learning

Computers can learn from data. Machine learning is a kind of statistical inference that finds patterns in data. Many areas of AI have progressed significantly in recent years thanks to learning algorithms that create new representations. For the approach to succeed, tremendous amounts of data are required. This "training data" must usually be supplied by people, but is sometimes acquired by the machine itself.

4. Natural Interaction

Intelligent agents require many kinds of knowlege to interact naturally with humans. Agents must be able to converse in human languages, recognize facial expressions and emotions, and draw upon knowledge of culture and social conventions to infer intentions from observed behavior. All of these are difficult problems. Today's AI systems can use language to a limited extent, but lack the general reasoning and conversational capabilities of even a child.

5. Societal Impact

AI can impact society in both positive and negative ways. AI technologies are changing the ways we work, travel, communicate, and care for each other. But we must be mindful of the harms that can potentially occur. For example, biases in the data used to train an AI system could lead to some people being less well served than others. Thus, it is important to discuss the impacts that AI is having on our society and develop criteria for the ethical design and deployment of AI-based systems.

Computers perceive the world using sensors.
1 - PERCEPTION
Object ID: Human
Accuracy: 99.4%
2 - REPRESENTATION & REASONING
Agents maintain representations of the world and use them for reasoning.
3 - LEARNING
Computers can learn from data.
4 - NATURAL INTERACTION
Intelligent agents require many kinds of knowledge to interact naturally with humans.
5 - SOCIETAL IMPACT
AI can impact society in both positive and negative ways.
AI4K12 .org

Figure 2-1: AI4K12's Five Big Ideas in AI.

	What students learn:	What students do:

K-2	• **Sensing in Living Things**: People experience the world through sight, hearing, touch, taste, and smell. • **Computer Sensors**: Computers "see" through video cameras and "hear" through microphones. • **Abstraction Pipeline, Language**: For a computer to understand speech it must be able to recognize the sounds from which words are constructed. • **Inclusivity**: Speech recognition systems need to accommodate different languages because many different types of people will use them.	• **Sensing in Living Things**: Identify human senses and sensory organs. • **Computer Sensors**: Locate and identify sensors (camera, microphone) on computers, phones, robots, and other devices. • **Sensing vs. Perception**: Give examples of intelligent vs. non-intelligent machines and explain what makes a machine intelligent. • **Abstraction Pipeline, Language**: Describe the different sounds that make up one's spoken language, and for every vowel sound, give a word containing that sound. • **Inclusivity**: Discuss why intelligent agents need to understand languages other than English.

3-5	• **Sensing in Living Things**: Some animals experience the world differently than people do. • **Digital Encoding**: Images are encoded as 2D arrays of pixels, where each pixel is a number indicating the brightness of that piece of an image, or an RGB value indicating the brightness of the red, blue, and green components. • **Feature Extraction**: Face detectors use special algorithms to look for eyes, noses, mouths, and jawlines. • **Abstraction Pipeline, Vision**: Understanding complex scenes requires taking into account the effects of occlusion when recognizing objects. • **Types of Domain Knowledge**: Speech recognition systems are trained on millions of utterances, which helps them select the most likely interpretation of the signal.	• **Sensing in Living Things**: Compare human and animal perception. • **Digital Encoding**: Explain how images are represented digitally in a computer. • **Feature Extraction**: Illustrate how face detection works by extracting facial features. • **Abstraction Pipeline, Vision**: Illustrate how outlines of partially occluded (blocked) objects differ from the real shapes of the objects. • **Types of Domain Knowledge**: Demonstrate how a text to speech system can resolve ambiguity based on context, and how its error rate goes up when given ungrammatical or meaningless inputs.

6-8	• **Digital Encoding**: Sounds are digitally encoded by sampling the waveform at discrete points (typically several thousand samples per second), yielding a series of numbers. • **Sensing vs. Perception**: There are specialized algorithms for face detection, facial expression recognition, object recognition, obstacle detection, speech recognition, vocal stress measurement, music recognition, etc. • **Feature Extraction**: Locations and orientations of edges in an image are features that can be extracted by looking for specific arrangements of light and dark pixels in a small (local) area. • **Abstraction Pipeline, Vision**: The progression from image to meaning takes place in stages, with increasingly complex features extracted at each stage.	• **Digital Encoding**: Explain how sounds are represented digitally in a computer. • **Sensing vs. Perception**: Give examples of different types of computer perception that extract meaning from sensory signals. • **Feature Extraction**: Illustrate the concept of feature extraction from images by simulating an edge detector. • **Abstraction Pipeline, Vision**: Describe how edge detectors can be composed to form more complex feature detectors for letters or shapes.

9-12	• **Computer Sensors**: Sensors are devices that measure physical phenomena such as light, sound, temperature, or pressure. • **Abstraction Pipeline, Language**: waveforms –> articulatory gestures –> sounds –> morphemes –> words –> phrases –> sentences • **Types of Domain Knowledge**: Domain knowledge in AI systems is often derived from statistics collected from millions of utterances or images. • **Inclusivity**: Dark or low contrast facial features are harder to recognize than high contrast features. Children's speech is in a higher register and less clearly articulated than adult speech.	• **Computer Sensors**: Describe the limitations and advantages of various types of computer sensors. • **Abstraction Pipeline, Language**: Illustrate the abstraction hierarchy for speech, from waveforms to sentences, showing how knowledge at each level is used to resolve ambiguities in the levels below. • **Types of Domain Knowledge**: Analyze one or more online image datasets and describe the information they provide and how this can be used to extract domain knowledge for a vision system. • **Inclusivity**: Describe some of the technical difficulties in making computer perception systems function for diverse groups.

Table 2.1: Knowledge items and learning objectives for AI4K12's Big Idea #1: Perception for K-12 grade bands [131].

AI4K12 has also released a more basic set of items for what students should be able to do for Big Idea #3, which corresponds to machine learning [37]. These are:

- "Grades K-2:
  - Learn from patterns in data with 'unplugged activities'

  - Use a classifier that recognizes drawings. Use Google Autodraw or Cognimates Train Doodle to investigate how training sets work to identify images and discuss how the program knows what they are drawing

- Grades 3-5:
  - Describe and compare the three different machine learning approaches: supervised, unsupervised and reinforcement learning.
  - Modify an interactive machine learning project by training its model.
  - Describe how algorithms and machine learning can exhibit biases.

- Grades 6-8:
  - Identify bias in a training data set and extend the training set to address the bias
  - Hand-simulate the training of a simple neural network

- Grades 9-12:
  - Train a neural network (1-3 layers) with TensorFlow Playground
  - Trace and experiment with a simple ML algorithm"

This work by AI4K12 is still in progress, with the most recent updates given in summer 2020. There has been much work in creating hands-on public ML tools for youth, short workshops and activities that help students understand AI and ML, and some longer curricula. The timeline for K-12 AI education is similar to that of general adult education; materials have just started coming out in the past 2-3 years.

2.2 Education Theories as Applied to ML and AI Education

I will now discuss the larger philosophical considerations from education that influenced my work. Education theories and research have a lot to say about how to effectively teach complex, technical subjects to general audiences. In general, in order for students to understand and become engaged in a difficult topic, it is essential that they feel empowered to create personally meaningful and useful artifacts. Self-efficacy theory deconstructs four elements that support this development. Constructionism and computational action identify ways to use a student's community and environment to support their learning. Experiential learning theory addressed the detailed processes of building hands-on activities into curricula.

2.2.1 Self-Efficacy

Self-efficacy, defined as a belief in one's competence or chances of successfully accomplishing a task and producing a favourable outcome, is a prominent social learning theory that is highly correlated to positive academic outcomes [7]. It was proposed by Albert Bandura in the 1900s and has since been closely tied to student success and persistence in educational settings. Research shows that self-efficacy beliefs shape how one approaches goals, tasks, challenges [77], and career aspirations [8]. It is important to note that self-efficacy as framed by Bandura is task-specific and does not relate to general feelings of confidence that are not task related.

There are four components of self-efficacy: enactive mastery, vicarious experiences, persuasion and support, and physiological experiences (see Figure 2-2) [7, 6]. Enactive mastery experiences arise from having the student directly and successfully perform tasks related to the learning objectives. Vicarious experiences refers to how people can develop and strengthen their sense of self-efficacy through seeing others like themselves succeed. Persuasion and support leads to self-efficacy when students feel encouraged by feedback from respected friends and advisors relating to the educational goals. Emotional and physiological experiences also influence students' interpretations of their own abilities. During educational interventions, feelings such as stress, anxiety, and shame may have dampening effects on self-efficacy while feelings such as wonder, joy, or excitement may increase self-efficacy.

Research suggests that the most important component of developing a strong sense of self-efficacy is enactive mastery [6]. When creating ML and AI curricula,

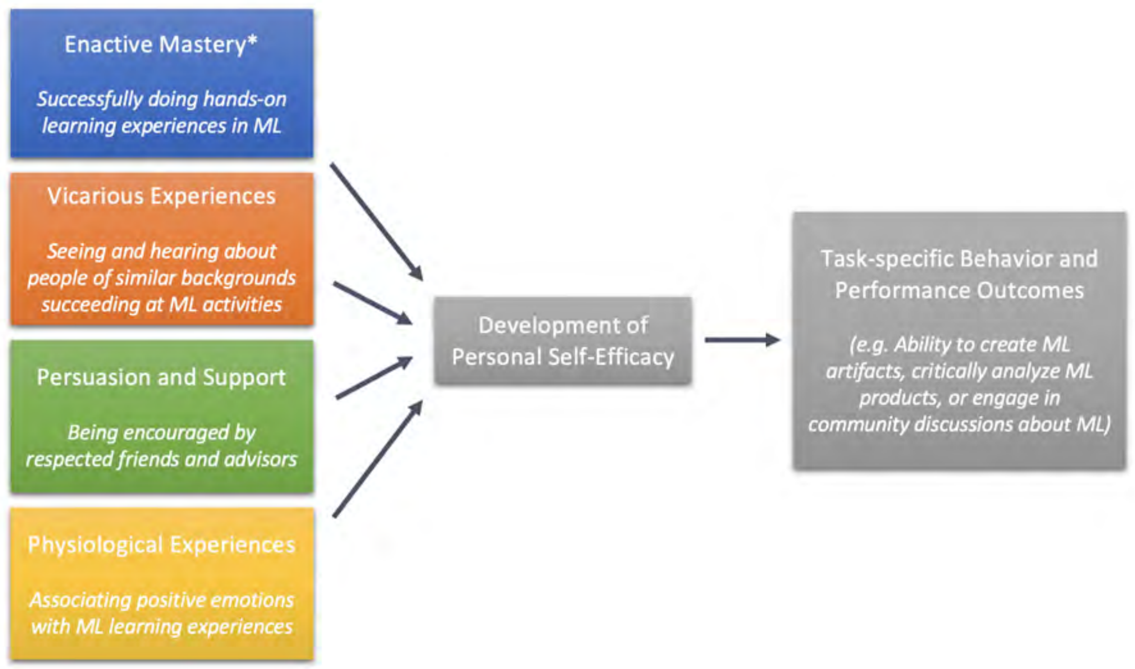

Enactive Mastery*
Successfully doing hands-on learning experiences in ML
Vicarious Experiences
Seeing and hearing about people of similar backgrounds succeeding at ML activities
Persuasion and Support
Being encouraged by respected friends and advisors
Physiological Experiences
Associating positive emotions with ML learning experiences
Development of Personal Self-Efficacy
Task-specific Behavior and Performance Outcomes
(e.g. Ability to create ML artifacts, critically analyze ML products, or engage in community discussions about ML)

Figure 2-2: The theory of self-efficacy as applied to machine learning education. Enactive mastery is the most important component in developing self-efficacy.

it will be important to design hands-on learning experiences that are simultaneously challenging and set students up for success. This will require educators to carefully consider the available tools for ML artifact creation and exploration, and may even motivate new tools to be built that fulfill instructional design needs. In practice, when educating students from a wide range of backgrounds, educators may also wish to emphasize collaborative work, work with near-peer mentors, and exposing students to ML professionals of diverse demographics (gender, age, and race/ethnicity) as means to build self-efficacy.

2.2.2 Constructionism, Computational Action, and Computational Identity

Constructionism is basically, as Seymour Papert stated, the idea that "children learn best when they are actively engaged in constructing something that has personal meaning to them" [100]. In line with constructionism, computational action encompasses the idea that when teaching computing specifically, we should start with authentic, real world applications instead of technical concepts and processes [129]. In computational action, to make computing education more inclusive, motivating, and empowering, students should have opportunities to create with computing that have direct impact on their lives and their communities. There are two components of computational action: computational identity and digital empowerment. It is important to make sure that students develop "computational identity", which is "seeing oneself as able to design and implement computational artifacts and also seeing oneself as as a part of the computing creator community" [129]. This definition is analogous to the construct of "scientific identity" as it appears in science education literature [133]. Formation of computational identity connects personal learning to community spheres and also encourages self-directed learning after the course is over. Digital empowerment refers to instilling the belief that students "can put their computational identity into action in authentic and meaningful ways on issues that matter to them".

Development of computational identity and computational action are closely tied to the "persuasion support" component of self-efficacy. In order to foster computational identity, students need to both succeed at computational tasks and engage in positive experiences as members of the relevant technical communities. Computational action requires students to participate in meaningful projects in service of their local or global communities, which will hopefully evoke positive messages of support from others and increase interest in the subject. Interwoven with approaches to deepen engagement and the development of self-efficacy, computational identity and computational action address how learners can connect their personal learning to community spheres. Computational action also emphasizes that instead of "generic" examples of applications, students should be interacting with "authentic" applications that directly have impact on their real lives.

In the context of machine learning, computational action is built as learners begin to see themselves as practicing machine learning creators while engaged in authentic ML practices as part of an ML community. It can be elicited through learning experiences where students apply ML to real issues and problems that they care about. Students' resulting projects will further engage them in learning and wanting to continue in the practice, forming a virtuous cycle capable of reinforcing and rewarding student behavior. When we apply these ideas to larger frameworks of ML education, we want students to form personal identities in relation to ML such that they feel like they can design and create personally meaningful ML applications to influence their world and identify as members of a larger community of ML creators.

2.2.3 Experiential Learning Approaches as Applied to ML

Experiential learning is defined as "learning through reflection and doing" and is founded in the work of Dewey (1938) [25], Lewin (1951) [74], Piaget and Inhelder (1969) [106], and Kolb (1984, 2014) [66, 67]. It is also related to Papert's work on constructionism, as it is elicited through direct manipulation of objects or systems as "objects to think with" [100]. Experiential learning approaches have been shown to be particularly suitable for use in computationally-rich learning environments. Laboratory experiments, including those using remote, virtual and hands-on laboratories, have been seen to lead to more efficient learning than passive reading or watching [2].

Kolb (1984) described four main phases of knowledge construction in his experiential learning theory: concrete experience (doing or having an experience), reflective observation (reviewing and reflecting on the experience), abstract conceptualization (concluding or summarizing learning from the experience), and active experimentation (planning or trying out new experiments based on what you have learned) [66]. These four phases follow a perception/processing continuum (see Figure 2-3), and form an iterative cycle during which meaningful learning occurs through many rounds of direct manipulation followed by reflection [47]. It is important to note that in the context of machine learning education, direct manipulation includes not only running demonstrations of the machine learning systems but also making modifications to the systems (through code or other means) and testing the results of the modifications in real-time.

These knowledge construction approaches are traditionally applied to machine learning education by using programming tools such as Python or R as the cornerstones of the "active experimentation" stage. However, as mentioned previously, these

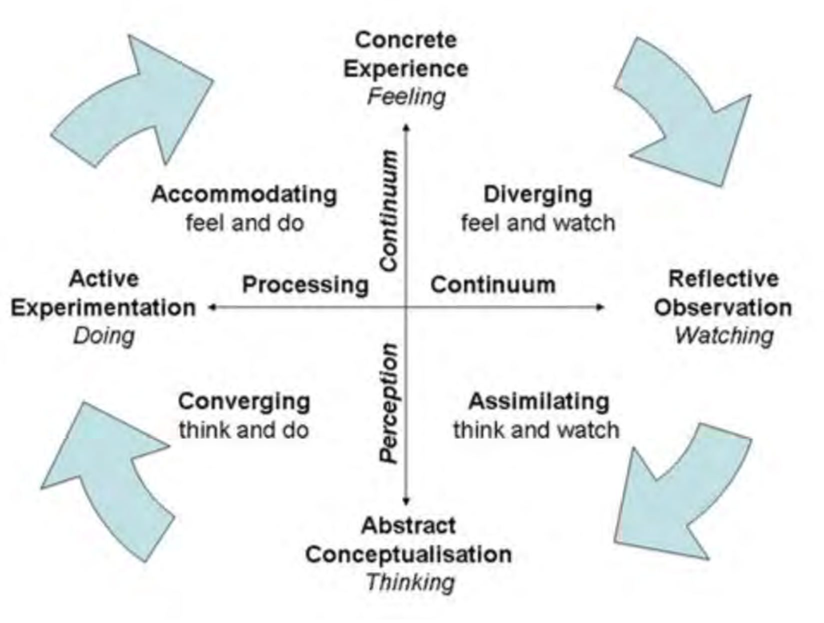


Figure 2-3: Kolb's four-stage experiential learning cycle [66].

tools present high barriers to entry, halting many potential learners before they are able to complete the first phase. Additionally, programming tools may slow down learners who are trying to quickly build up their masteries of ML by traversing the experiential learning cycle over multiple iterations — even students with adequate programming experience can spend a lot of time debugging simple errors when working with a new machine learning library that they are unfamiliar with.

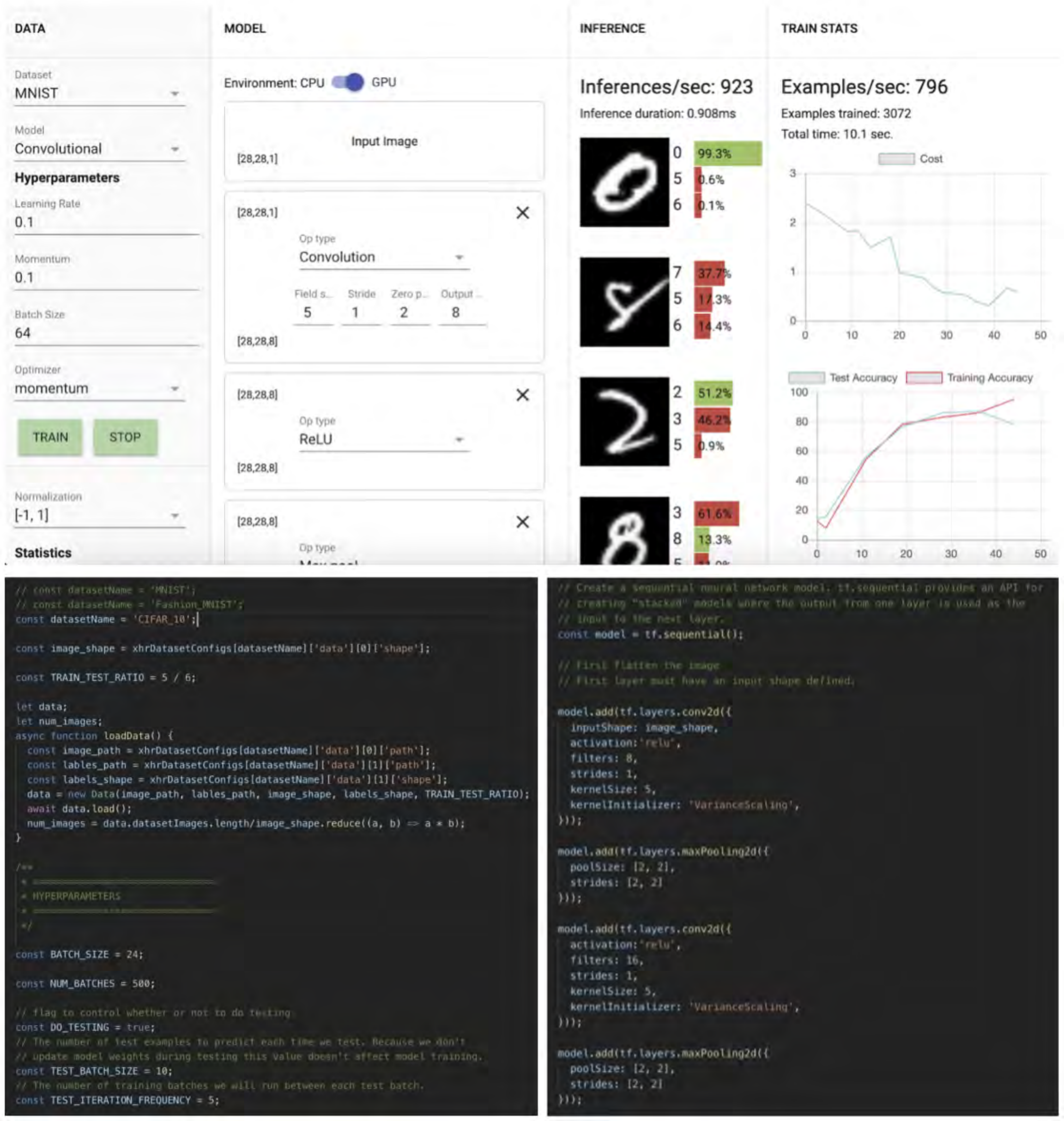
Fortunately, many companies and organizations are starting to build user-friendly applications for exploring and creating machine learning models. Web applications such as Teachable Machine [118], Model Builder [55], and Embedding Projector [125] enable students to directly manipulate machine learning systems to rapidly gain experience. Such user interfaces often entirely shield students from having to deal with syntax errors, and even cover some compiler and logic errors as well. Figure 2-4 shows the difference between using Model Builder, a user-friendly neural network training and analysis tool, and writing TensorFlow.js code that achieves similar learning objectives. These abstractions can be very valuable for the initial stages of learning in removing potential avenues of frustration with the learning experience that have little to do with the concepts behind the topic at hand. The existence of these technologies has also made it possible to teach machine learning on in-browser platforms that don't require high levels of computational power. Educators can use such public ML tools to develop learning experiences that fully support direct manipulation without the constraints that existed previously. Furthermore, such "laboratory" experiences with ML systems would provide students with experiences that directly relate to the future of work at the human-machine frontier.

Of course, once students use these public ML tools to gain enough general ML knowledge and method-specific knowledge after sufficient experiential learning cycles, they may wish to continue their learning trajectories by delving deeply into code. Good public ML tools will support this transition superbly, as they will have already exposed students to key concepts, terms, and architectural structures necessary for coding machine learning models. Some of that knowledge will directly transfer over as students learn how to program using ML libraries.

These four education theories are the foundations upon which my ML Education Framework is built, since my goal is to try to engage more people in ML so that they can understand how ML products work, can use them effectively and safely, and can participate in discussions around policies and regulations. The main takeaways that I use in my research philosophy are that: (1) students need to be experiencing hands-on learning with ML; (2) they need to create personally meaningful ML artifacts; (3) they need to be able to identify extended learning communities and connect with those communities; and (4) we need to teach using unintimidating, non-frustrating tools that are appropriate for the audience's background and level.

2.3 Similar Educational Frameworks in Computing and Engineering

As we design curricula and learning experiences for teaching AI and ML, it is important to consider related research into good pedagogical practices. Before I present my ML Education Framework, I will discuss two frameworks for computing and en-



Dataset
MNIST

Model
Convolutional

Hyperparameters

Learning Rate
0.1

Momentum
0.1

Batch Size
64

Optimizer
momentum

Normalization
[-1, 1]

Op type
Convolution

Field s...	Stride	Zero p...	Output ...
5	1	2	8

Op type
ReLU

```
// const datasetName = 'MNIST';
// const datasetName = 'Fashion_MNIST';
const datasetName = 'CIFAR_10';

const image_shape = xhrDatasetConfigs[datasetName]['data'][0]['shape'];

const TRAIN_TEST_RATIO = 5 / 6;

let data;
let num_images;
async function loadData() {
  const image_path = xhrDatasetConfigs[datasetName]['data'][0]['path'];
  const lables_path = xhrDatasetConfigs[datasetName]['data'][1]['path'];
  const labels_shape = xhrDatasetConfigs[datasetName]['data'][1]['shape'];
  data = new Data(image_path, lables_path, image_shape, labels_shape, TRAIN_TEST_RATIO);
  await data.load();
  num_images = data.datasetImages.length/image_shape.reduce((a, b) => a * b);
}

/**
 * ===============================
 * HYPERPARAMETERS
 * ===============================
 */

const BATCH_SIZE = 24;

const NUM_BATCHES = 500;

// flag to control whether or not to do testing
const DO_TESTING = true;
// The number of test examples to predict each time we test. Because we don't
// update model weights during testing this value doesn't affect model training.
const TEST_BATCH_SIZE = 10;
// The number of training batches we will run between each test batch.
const TEST_ITERATION_FREQUENCY = 5;
```

```
// Create a sequential neural network model. tf.sequential provides an API for
// creating "stacked" models where the output from one layer is used as the
// input to the next layer.
const model = tf.sequential();

// First flatten the image
// First layer must have an input shape defined.

model.add(tf.layers.conv2d({
  inputShape: image_shape,
  activation:'relu',
  filters: 8,
  strides: 1,
  kernelSize: 5,
  kernelInitializer: 'VarianceScaling',
}));

model.add(tf.layers.maxPooling2d({
  poolSize: [2, 2],
  strides: [2, 2]
}));

model.add(tf.layers.conv2d({
  activation:'relu',
  filters: 16,
  strides: 1,
  kernelSize: 5,
  kernelInitializer: 'VarianceScaling',
}));

model.add(tf.layers.maxPooling2d({
  poolSize: [2, 2],
  strides: [2, 2]
}));
```

Figure 2-4: Creating and training a convolutional neural network with Model Builder (top) [55] vs. through code (bottom).

gineering education that are built upon similar theories in education: computational thinking and CDIO.

2.3.1 Computational Thinking

Computational thinking is the method of framing a problem and problem-solving that arises from the core principles of computing [73]. Aside from its influence on society through computer science, computational thinking has led researchers to create innovative solutions for challenges from sequencing the human genome to microeconomics [137]. In 2012, Brennan and Resnick established a framework for studying and assessing the development of computational thinking. Through their research with Scratch, a blocks-based programming language, they proposed a three-pronged approach to considering computational thinking through: (1) concepts, which refer to technical programming concepts; (2) practices, or good programming methodology; and (3) perspectives, which describe how an individual relates to the technological world [14]. Over the past decade, computational thinking has gained funding and traction in primary education internationally due to the emergence of a generation more reliant on technology than ever before [41]. Despite initial skepticism, researchers, and teachers have also shown that it is both possible and valuable to teach students computational thinking at a young age.

Two large surveys of computational thinking education conducted in the past decade have gained much traction: In Lee et. al.'s 2011 paper detailing a set of NSF-supported STEM programs, it was found that existing definitions of computational thinking can be applied to K-12 settings for students from a wide range of technical and socioeconomic backgrounds [73]. To model and understand the specific values that students took away from computational thinking lessons, they developed a three-prong framework: (1) Abstraction, applying concepts from one use case to another; (2) Automation, using a computer to execute a set of repetitive tasks more efficiently than a human; and (3) Analysis, a reflective process of the students' assumptions and implementations. More recently, Duncan and Bell analyzed a variety of English, Australian, and CSTA curricula that had been published for primary schools [29].

They established the main topics covered in all of these curricula and the suitability of the material for first year to eighth year primary school students. They also emphasized the importance of teacher development—once teachers understood why computational skills were important to teach, they were more comfortable and confident with teaching the material. One key takeaway from this study was that the content of most of the curricula they analyzed focused on programming skills and data representation knowledge instead of developing more intrinsic perspectives, such as empowering students to be technological creators. Researchers had also hoped that computational thinking skills would be taught indirectly via the learning of computer skills, but found that this was generally not the case.

Grover and Pea have also conducted a pedagogical study of methods for introducing computational thinking concepts in middle schools [41]. Instead of teaching through traditional text-based code, they chose to use MIT App Inventor, a blocks-based programming tool. MIT App Inventor has the goal of allowing people with little to no programming experience to make highly functional mobile applications, and it is particularly well-suited to empowering beginner audiences in creating useful technical artifacts [56]. Based on the workshops they ran, they concluded that there should be a much heavier emphasis on social interaction in the development of individual mental processes in computational thinking education. Specifically, there is a need to develop communicative activities during classes so that students can foster better individual and group learning. Grover and Pea felt that ideas of classroom discourse should be brought in from the learning sciences to teach introductory computational concepts. In concert with computationally rich activities, discussions and reflection sessions with peers significantly influenced the organic introduction and use of new vocabulary as well as important foundational computing concepts that learners were previously unaware of.

It has been said that AI and ML education are the next stages of computational thinking education. There are many relevant lessons that we can take from the development of the computational thinking education field as we move into similar territory with respect to machine learning. First, it is entirely possible to teach complex technical concepts to children and general audiences in meaningful, useful ways. Applicable technical abstractions need to be made when establishing the knowledge, skills, and attitude goals during the design of educational interventions. Tools that are both empowering and at an appropriate level for the audience need to be identified and used. Teachers need to be properly supported so that they have clear understandings of good pedagogical approaches for the technical topics at hand. It will be useful to keep these foundational educational frameworks in mind as we move to pioneer similar areas for machine learning education targeted towards youth and general audiences.

2.3.2 Conceiving - Designing - Implementing - Operating (CDIO) and Use-Modify-Create (UMC)

Conceiving — Designing — Implementing — Operating (CDIO) is a prominent educational framework used in general engineering education, specifically at the university and professional levels [20]. The first CDIO syllabus was created in 2001 with the purpose of codifying a comprehensive understanding of skills needed for professional engineers to excel in industry (i.e. technical, personal, interpersonal, and system-building skills) and defining new educational approaches to engineering education based on these real-world skills [19]. The most recent CDIO syllabus was released in 2011 and presents three building blocks of "knowledge", "skills", and "attitudes" that support engineering students' abilities to "Conceive, Design, Implement, and Operate Systems in the Enterprise, Societal and Environmental Context" [20]. Within this syllabus, knowledge is developed first, followed by professional skills and personal attitudes, then interpersonal skills, and finally the integrated CDIO process. Table 2.2 presents items from the syllabus at a second level of detail.

Researchers and educators are starting to apply CDIO principles to artificial intelligence and machine learning engineering education in areas including medical therapy and robotics [79, 10]. More research needs to be done to determine exactly how machine learning will fit into the CDIO framework, but the backbone of "knowl-

1. Disciplinary Knowledge and Reasoning 1.1 Knowledge of underlying mathematics and science 1.2 Core fundamental knowledge of engineering 1.3 Advanced engineering fundamental knowledge, methods and tools
2. Personal and Professional Skills and Attributes 2.1 Analytical reasoning and problem solving 2.2 Experimentation, investigation and knowledge discovery 2.3 System thinking 2.4 Sttitudes, thought and learning 2.5 Ethics, equity and other responsibilities
3. Interpersonal Skills: Teamwork and Communication 3.1 Teamwork 3.2 Communications 3.3 Communications in foreign languages
4. Conceiving, Designing, Implementing, and Operating Systems in the Enterprise, Societal and Environmental Context 4.1 External, societal and environmental context 4.2 Enterprise and business context 4.3 Conceiving, systems engineering and management 4.4 Designing 4.5 Implementing 4.6 Operating

Table 2.2: The CDIO syllabus for university and professional level engineering education at a second level of detail [20].

edge", "skills", and "attitudes" in support of professional engineering practices seems transferable to this field. It is interesting to note that due to the importance of feedback loops in ML systems that are constantly learning from live data and updating, a CDIO framework tailored to ML may want to establish a more prominent piece within the "Design" part of the syllabus that educates students on implementing feedback/continuous evaluation processes in ML systems.

While CDIO has been traditionally targeted towards students on a technical career trajectory, Use-Modify-Create (UMC) is a lesson progression that has been shown to support youths' development and engagement in computational thinking [73]. In a variety of programs for youth, UMC has engaged learners in progressively deeper computational thinking experiences within rich computational environments. This pattern of engagement is based on the premise that scaffolding increasingly deeper interactions will promote the acquisition and development of computational thinking (see Figure 2-5). In the "Use" stage, learners are initially consumers of someone else's creation. Over time, in the "Modify" phase, they begin to alter the model or system with increasing levels of sophistication. As learners gain skills and confidence, they are encouraged to develop ideas for new computational models or systems of their own design that address issues of their own choosing. Within this "Create" stage, all three key aspects of computational thinking — abstraction, automation and analysis — come into play. UMC draws elements from Kolb's Experiential Learning Theory (1984) [66] wherein knowledge is created through the transformation of experience and Vygotsky's Social Constructivism (1978) [134], which posits that students construct their own knowledge from experiences that have personal meaning to them through discussion and collaboration.

The Use-Modify-Create progression has been applied to machine learning education to help students deepen their understanding of ML concepts and gain mastery of practical ML skills [71]. First, students use ready-made ML models within fast-response and user-friendly interfaces to develop high-level intuitions about training, testing, and the importance of data. Then students manipulate the models directly through modifying code to gain understanding of how different architectures, hyper-

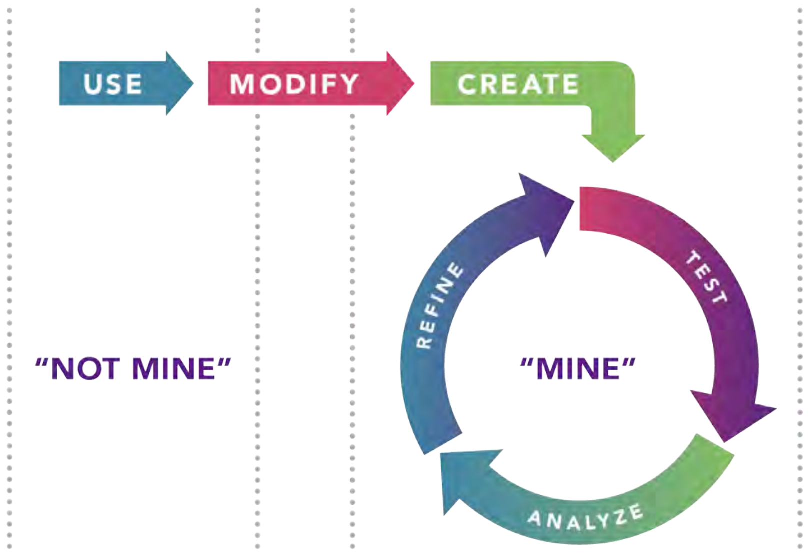


Figure 2-5: A diagram of the Use-Modify-Create learning progression proposed in [73] as adapted by K12CS.org [35].

parameters, and data impact results for various types of problems. Finally, students scope out a problem suitable for ML and create an ML application of their own.

In considering the CDIO framework and the UMC progression in the context of machine learning education, both approaches may be suitable for students of different backgrounds. CDIO appears to be appropriate for educating ML developers and engineers while UMC may be more appropriate for general education, as we consider transitioning ML consumers to ML tinkerers, critical consumers, and ML-engaged citizens. Unlike most traditional engineering fields, machine learning can be highly nondeterministic and uninterpretable. The hands-on-first approach of the Use-Modify-Create progression can help general audiences without technical backgrounds quickly develop useful intuitions about these complex technologies. We can also consider using UMC-centric curricula as precursors to CDIO curricula in transitioning students who have completed a general ML education who wish to further pursue a more technical, career-focused ML education.

The course that I will discuss in Chapter 4 (6.S198: Deep Learning Practicum) is an example of how the Use-Modify-Create progression can be successfully applied to teach machine learning to general audiences who may not have backgrounds in computer science.

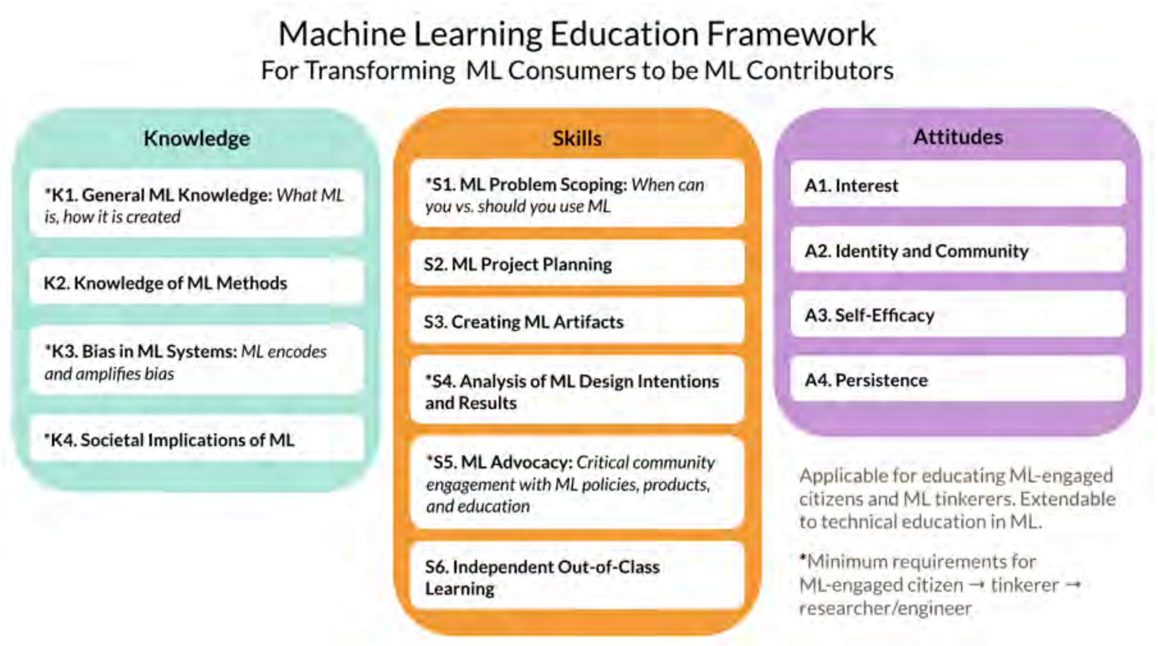
3

The Machine Learning Education Framework

In this chapter, I introduce my framework for ML education targeting a tinkerer/consumer audience. In Chapter 4, I will present a machine learning course for general adult audiences that was built using this framework, and in Chapter 5, I will analyze 6 ML and AI courses for various audiences using a curriculum rubric based off of this framework.

The Machine Learning Education Framework was inspired by educational theories of self-efficacy, constructionism, and computational action. It clarifies the knowledge and skills that are most essential for the general public and is organized into three main sections: what concepts students need to know about ML, what skills they need to have, and what attitudes we want them to develop towards ML. It also has a similar organization as frameworks for computational thinking and CDIO.

The three top level categories of the framework were chosen to encompass the holistic process of teaching people ML and ensuring continued participation in the field. “Knowledge” about ML will be necessary for individuals to thrive in today’s ML-driven world and builds foundations for further engagement with the technology. ML “Skills” define specific actions and activities that students need to do and practice in order to become ML contributors. The category "Attitudes" reflects long-term societal goals for educational initiatives in ML, and addresses the perspectives that



Machine Learning Education Framework
For Transforming ML Consumers to be ML Contributors
Knowledge
*K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
*K3. Bias in ML Systems: ML encodes and amplifies bias
*K4. Societal Implications of ML
Skills
*S1. ML Problem Scoping: When can you vs. should you use ML
S2. ML Project Planning
S3. Creating ML Artifacts
*S4. Analysis of ML Design Intentions and Results
*S5. ML Advocacy: Critical community engagement with ML policies, products, and education
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence
Applicable for educating ML-engaged citizens and ML tinkerers. Extendable to technical education in ML.
*Minimum requirements for ML-engaged citizen → tinkerer → researcher/engineer

Figure 3-1: The Machine Learning Education Framework.

students should develop through an ML course.

While this framework was created to specifically be applicable for transforming ML consumers into ML contributors who are civically engaged and able to tinker with ML, this framework is also extendable to further technical education in ML.

3.1 The Machine Learning Education Framework

A summary diagram of the framework is presented in Figure 3-1. There are 4 knowledge items, 6 skills, and 4 attitudinal items. In this section, I will detail each item and describe how these elements are interconnected.

3.1.1 Knowledge

K1, the first knowledge item, is General ML Knowledge. This refers to knowing what machine learning is, and understanding the entire pipeline of how ML systems are created, from the first steps of problem identification and scoping to data collection, training, testing, deployment, and setting up structures to constantly incorporating new data and feedback into improving the ML system.

K2 is Knowledge of ML methods. This refers to being able to accurately discern when to use a range of machine learning methods across the breadth of the field (e.g. k-nearest neighbors, CARTs or decision trees, neural networks, ensemble methods) and also understanding how different methods work.

K3 is Bias in ML Systems. The core idea behind this item is that just because ML algorithms are automated or math-based does not automatically make them neutral or fair. It is entirely possible for ML systems to create as biased outcomes as humans decision-makers or even amplify bias. At deeper levels, students should know the ways that bias can be incorporated in each step of the machine learning pipeline and also have some understanding of how to incorporate practices of ethical thinking and design.

K4 is Societal Implications of ML. Students should know that ML systems can have widespread positive and negative impact when implemented in a society. They should also recognize that creators of ML technologies need to consider ethical, cultural, and societal implications of their work.

These 4 knowledge items encompass both technical concepts and ethical, cultural, and societal considerations relating to ML.

3.1.2 Skills

Under the skill category, the first skill, S1, is ML problem scoping. This concerns whether or not students are able to determine which problems *can* be solved using ML (i.e. if the problem is such that past data predicts future outcomes and if it's possible to obtain a lot of training data) and also which problems *should* be solved using ML from a logical, cultural, or ethical perspective. This is a skill that many students struggle with. It's fairly common for beginners to get the impression that ML can be used for absolutely anything, but, for example, it may not make sense to apply it to problems like writing healthcare legislation or predicting when a star will burn out.

S2 is ML Project Planning. Following from S1, when you are given a problem that is suitable for ML, are you able to plan out a solution sensitive to both technical implementation considerations (i.e. what data is needed, how to obtain the data, what resources are available for training, what metrics to use for analysis) and contextual factors such as how the system fits into existing human, technical, or systemic processes and considerations like data privacy.

S3 is Creating ML Artifacts. This skill covers whether or not students are able to use appropriate tools to create a range of machine learning artifacts. There are a wide range of appropriate tools for different audiences and different applications. For data collection for example, this skill can encompass both using a website like Kaggle or writing code to scrape data using a library such as Beautiful Soup. For application creation, tools can vary from PyTorch or TensorFlow to Cognimates or Teachable Machine.

S4 is Analysis of ML Design Intentions and Results. While S3 is the skill of creating technical ML artifacts, S4 encompasses the skill of creating analytical ML artifacts. Students need to be able to extract both implicit and explicit design intentions of an ML system and be able to critically analyze those initial intentions in relation to how the deployed system can and should be used (e.g. considering the affordances of a system, use cases, or accuracy targets of predictive classes in relation to socio-political intentions or financial incentives).

S5 is ML advocacy, which refers to students' abilities to critically engage with their communities regarding machine learning policies, products, and education.

S6 is Independent Out-of-Class Learning. Due to ML being a very rapidly developing field, it is important for students to be able to learn new ML concepts or skills independently of educational interventions in order to keep up. It is difficult to measure and keep tabs on whether students do this after a course is over, but during a course, it's entirely possible to build activities that encourage students to seek out learning experiences outside of the classroom. This can be anything from following the news or blogs about ML to experimenting with new ML products or methods.

These 6 skills encompass critically thinking about ML, creating technical ML applications, engaging with ML on a societal level, and skills for personal future development in ML.

3.1.3 Attitudes

Under the Attitude category, the first item, A1, is interest. We want students to feel that ML is interesting and important to their lives so that they become highly motivated to engage more with the subject.

A2 is Identity and Community, which stems from the educational theories of computational action and identity. It is important for students to feel that they belong to a larger ML learning community where they can receive help from and give back to. During an educational intervention, the students can be encouraged to practice participating in both small learning communities of their peers and larger communities that may be online.

A3 is Self-Efficacy. We want students to feel fully empowered to engage more with machine learning, and design and build new and meaningful things — this can be technical applications or socio-analytical artifacts.

Finally, A4 is Persistence. This refers to whether or not students voluntarily and successfully pursue more and more experiences with machine learning in the future. For example, if they start keeping up with news about ML, take more classes in ML, complete new ML projects, or even pursue a career change to work with machine learning. This item can be very difficult and very resource-intensive to measure since it involves keeping track of a lot of people over time.

3.1.4 Essential Items of the Framework

There are three knowledge items and three skills that are minimally necessary for courses for ML-engaged citizens (starred * in the framework per Figure 3-1). These are K1. General ML Knowledge, K3. Bias in ML Systems, K4. Societal Implications of ML, S1. ML Problem Scoping, S4. Analysis of ML Design Intentions and Results, and S5. ML Advocacy. In many classrooms across the world, there may not be sufficient or available resources to run hands-on activities for technical artifact creation. In these cases, the 6 starred items are still possible to cover such that students become able to petition their government to address societal issues relating to ML or make

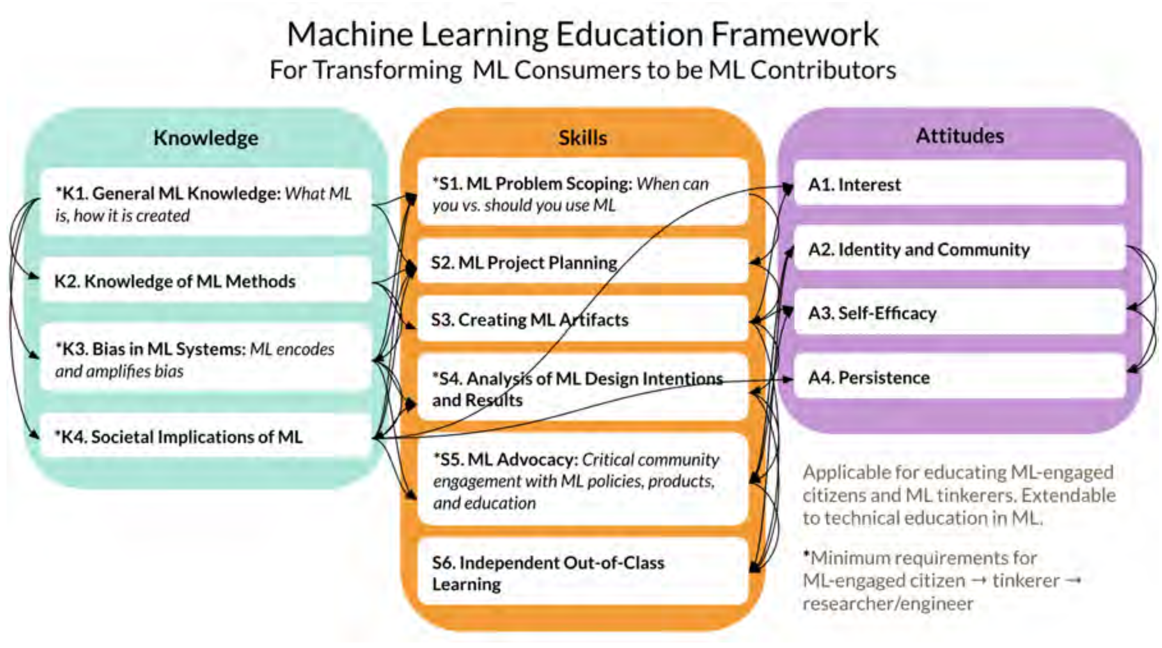

Machine Learning Education Framework
For Transforming ML Consumers to be ML Contributors
Knowledge
*K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
*K3. Bias in ML Systems: ML encodes and amplifies bias
*K4. Societal Implications of ML
Skills
*S1. ML Problem Scoping: When can you vs. should you use ML
S2. ML Project Planning
S3. Creating ML Artifacts
*S4. Analysis of ML Design Intentions and Results
*S5. ML Advocacy: Critical community engagement with ML policies, products, and education
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence
Applicable for educating ML-engaged citizens and ML tinkerers. Extendable to technical education in ML.
*Minimum requirements for ML-engaged citizen → tinkerer → researcher/engineer

Figure 3-2: Key linkages between elements of the ML Education Framework.

decisions about what ML technologies to use and how to use them in their personal lives.

These starred items should still be important to cover at some level in courses for ML tinkerers and even for technical courses meant for researchers and engineers.

3.2 Interconnections within the Framework

The 14 elements of this framework are closely interconnected (see Figure 3-2). In this section, I will detail the reasoning behind the interconnections by discussing which items feed into each element.

Within the Knowledge category, linkages are fairly intuitive. K1. General ML Knowledge is the initial root of the framework and is thus linked to every element in the framework. In these linkages, I will be highlighting the items for which strong ability in K1 is necessary. K1 aids in the development of K2. Knowledge of ML Methods.

In order to achieve K3. Bias in ML Systems, a student needs knowledge of the ML creation process to understand how bias can unwittingly occur in every step of the pipeline. Knowledge of specific ML methods (e.g. specific loss functions) also aid in developing a better technical understanding of how bias can occur and be mediated.

In order to achieve K4. Societal Implications of ML, students need both K1. General ML Knowledge, and K3. Bias in ML Systems. Understanding bias helps to motivate a deeper understanding of the huge societal impact that ML systems can have, both the good and the bad.

Three knowledge items feed into S1. ML Problem Scoping: K1. General ML Knowledge, K3. Bias in ML Systems, and K4. Societal Implications of ML. K1 helps the student answer whether or not a problem can be solved using ML, whereas K3 and K4 help the student think about whether or not a problem should be solved using ML.

All 4 knowledge items as well as S1 feed into S2. ML Project Planning. In order to create a detailed plan for a given ML project, the student needs holistic appreciation for both the technical and socio-ethical aspects of the ML problem.

S3. Creating ML Artifacts follows from K2. Knowledge of ML Methods and S2. ML Project Planning. The students need to understand the process of creating the artifact they are planning to make and also be able to use the appropriate tools and methods to do so.

S4. Analysis of ML Design Intentions and Results feeds from K3. Bias in ML Systems, K4. Societal Implications of ML, and S3. Creating ML Artifacts. In order for a student to properly analyze the effects of an ML system, they need to be able to consider the societal implications and potential ethical concerns towards the performance of the system. It is also valuable for the student to have knowledge of how the ML system was created so that they are able to make reasonable technical judgments and recommendations for improvement.

S5. ML Advocacy feeds from K3. Bias in ML Systems, K4. Societal Implications of ML, and S4. Analysis of ML Design Intentions and Results. In order to motivate people to participate in activism relating to ML, it is important that they know of the known ethical problems and potential widespread benefits and harms. Being able to critically analyze ML technologies in a professional manner will help people engage in more effective and convincing advocacy.

S6. Independent Out-of-Class Learning arises from S3. Creating ML Artifacts, S4. Analysis of Design Intentions and Results, S5. ML Advocacy, and A2. Identity and Community. Students are inspired to learn independently when they are highly motivated and engaged in creating ML artifacts (either technical or analytical). In order for students to succeed at independent out-of-class learning, they need to have a community that they can learn from and enough confidence in their ability to learn the material.

The two items that most contribute to developing A1. Interest in ML are: teaching them about the huge potential for ML to enact widespread change for good and for bad (K4. Societal Implications of ML), and transforming them from passive consumers of ML to active creators of it (S3. Creating ML Artifacts).

Two elements that feed heavily into A2. Identity and Community are S5. ML Advocacy and S6. Independent Out-of-Class Learning. In a bit of an obvious feedback loop, engaging in ML advocacy in the community helps students start the cycle of actively participating and molding an ML learning community, and the behaviors occur when students begin learning outside of class in an external community.

As previously discussed, self-efficacy has 4 components: enactive mastery, vicarious experiences, persuasion and support, and physiological states. Enactive mastery is developed when students create ML artifacts (S3) and when they engage in ML advocacy as a form of creating ideas, conversations, and change in their community (S5). Both vicarious experiences and persuasion and support come from A2. Identity and Community, where students can see their peers succeeding at doing ML tasks and receive positive feedback from their learning community. Physiological states are not directly addressed in the framework but emotions like nervousness or stress can be controlled in classic academic settings through good pedagogical practices and instructional design (i.e. being clear about prerequisites, properly supporting students through problem areas).

Finally, 3 other elements feed into A4. Persistence. As we consider how the future of work is expected to be closely tied to advances in machine learning, many people and organizations are interested in developing persistence in ML through entry-level ML courses. Specifically, when a student completes a course, we want them to consider pursuing more advanced learning in the field, or even start a career in ML some day. Research into persistence in education shows that there are three key components [128]: (1) whether or not the student believes in their ability to succeed (which comes from A3. Self-Efficacy), (2) whether or not the student feels a sense of belonging in the ML community (which comes from A2. Identity and Community), and (3) what the student perceives the value of the material to be. This third item can refer to recognizing societal implications or personal implications of ML (e.g. acknowledging that machine learning will be important to know and apply in order for students to safely and effectively navigate the world). This third item is closely related to K4. Societal Implications of ML.

In Chapter 5, I will analyze how other courses relate to this framework, as well as compare my ML Education Framework to the Five Big Ideas from the AI4K12 Initiative, which is another prominent set of guidelines for AI education in the United States.

4

6.S198: Designing a College-Level Actionable Machine Learning Practicum

"Before this course, I thought of computer programs more linearly – as if computer programmers were mostly in control of a program's results. Now I have a much greater understanding of how machine learning programs can be biased and unfair... I learned the importance of providing good, varied input data and how this data can have a significant impact on a network and ultimately the world."

– 6.S198 Student, Spring 2018

In this chapter, I present the curriculum, research results, and pedagogical findings for an actionable deep learning course I designed and taught at the Massachusetts Institute of Technology (MIT) using the Machine Learning Education Framework. This course was called "6.S198: Deep Learning Practicum" and had the goal of helping students develop self-efficacy in machine learning, specifically focusing on deep learning techniques. Following from that goal, the course aimed to help students with or without prior programming or AI/ML knowledge develop the knowledge and skills necessary to apply ML to interesting and suitable problems of their own choosing.

Instead of aiming to develop students into ML researchers or engineers, the course focused on the "doing and use of machine learning" and the creation of ML applications. As a practicum, the course was designed to incorporate mechanisms for engagement: empowering students, creating meaningful experiences through scaffolded, inquiry-based learning, and authentic learning opportunities.

The Deep Learning Practicum was offered at MIT in the spring and fall 2018 semesters. The course met twice a week for 1.5 hours each session over 15 weeks and offered a hands-on, application-based educational approach to ML. The course was 12 units, which is standard for a full semester course at MIT, and was designed as a small scale, lab-based experimental course that counted as either an Elective or an Independent Inquiry in EECS. The course was open to students from any major and grade level, with no requirements for prior programming or ML knowledge. The first offering in spring 2018 acted as a pilot that informed better pedagogical design for the next offering of the course. The instructors hoped that within a semester, the course would allow any student to gain the skills and knowledge necessary to apply machine learning to their work or daily lives. Most of the material from the two offerings of Deep Learning Practicum have been previously published [71, 72, 78].

4.1 Instruments and Assessment

Since the purpose of Deep Learning Practicum was to empower students to create with ML, the most important measurements we wanted to make for this research were of student "self-efficacy". When we taught the course, there were no validated instruments for measuring self-efficacy in ML, so we created a post-course survey instrument based on validated instruments for measuring self-efficacy in general sciences, including Children's Science Curiosity Scale (CSCS) [46] and Modified Attitudes Towards Science Inventory (mATSI) [135]. These references have been reliability validated across gender, ethnicity, and income levels. The CSCS has an internal consistency of .85 and test-retest reliability of .67-.69, and the mATSI demonstrates .70 reliability using Cronbach's alpha [51]. We also conducted qualitative assessments of student capstone projects using a top-down thematic approach for ML concepts and a bottom-up emergent approach for ML skills.

Students consented to have their outputs from the course be used for research purposes. The data sources used to determine the impact of the framework included anonymous responses to pre- and post- course surveys, analysis of weekly assignment grades, and analysis of capstone projects. The post-course survey had 12 Likert scale items, 3 multiple answer/multiple choice items, and 3 free response items. These questions were across three scales: self-efficacy, computational identity, and computational action. The scale for self-efficacy was divided into four subscales corresponding to components of self-efficacy described by Bandura: mastery experiences, vicarious experiences, persuasion/supportive messages, and physiological reactions. It was designed to take approximately 30 minutes.

4.2 Application of Educational Theories and the ML Education Framework

In the course design of Deep Learning Practicum, the key educational theories of self-efficacy, constructionism, computational action, and experiential learning (previously detailed in Chapter 2) were prominently employed to increase the accessibility of ML technologies to a wider, less technical college audience. Items from the ML Education Framework were also carefully built into the curriculum to reinforce the goals of the course.

4.2.1 Experiential learning and Use-Modify-Create in Deep Learning Practicum

Experiential learning exercises are combined with a capstone project through ITEST's Use-Modify-Create progression. We posited that Use-Modify-Create can help students deepen their understanding of ML concepts and master practical skills. During the first portion of the course, students learned about a variety of deep learning modules (i.e. convolutional neural networks, generative adversarial networks, embeddings). These modules were taught through the Use-Modify-Create progression. First, students used ready-made ML models within fast-response and user-friendly interfaces to develop high-level intuitions about training, testing, and the importance of data. Then, students manipulated the models directly to understand how different architectures, hyperparameters, and datasets impact results for different problems. This led to having students complete coding exercises on TensorFlow.js where they coded the models they had previously explored using graphical user interfaces. The latter portion of the course was centered on student-led capstone projects — students scoped a problem suitable for ML and created their own ML applications.

I will describe activities from the convolutional neural network (CNN) module in the course to illustrate how experiential learning principles and Use-Modify-Create were applied to teach technical units. First, students listened to a short technical lecture (10-15 minutes) where they learned a bit about the filtering and mathematical processes of how convolutional layers worked. Then, students played around with a visualization webapp during the initial "Use" phase of the CNN module (see Figure 4-1) [44]. This webapp allowed students to draw numbers and see how a CNN that had been trained on MNIST classified those numbers. The CNN was pretty shallow (only 8 layers) and the webapp allowed students to visualize and unpack each layer of the neural network. Students could also click and hover over each cell to see its input, calculation, and output. Students experimented with different inputs and discussed what they saw in the layers with their peers to gain an understanding about the model and all of the different components of each layer. In lieu of digging deeply into the mathematics of convolutional layers, students discussed questions such as "What properties of the image may affect what filter sizes you should choose?". These conversations helped them get an intuition for how to design these models properly (e.g. image size and the pen's stroke size may affect how one would choose the number of filters or filter sizes).

Next, students used the Model Builder webapp to move from the "Use" phase to the "Modify" phase of their learning progression [55]. Model Builder is a public

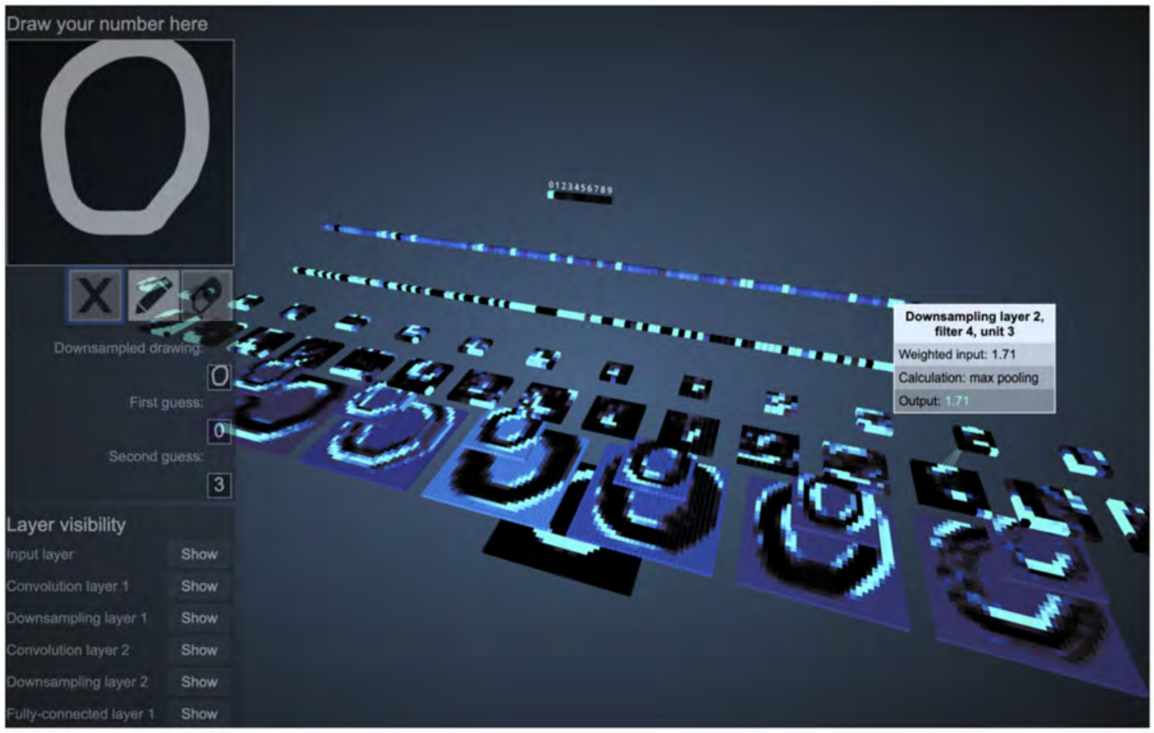
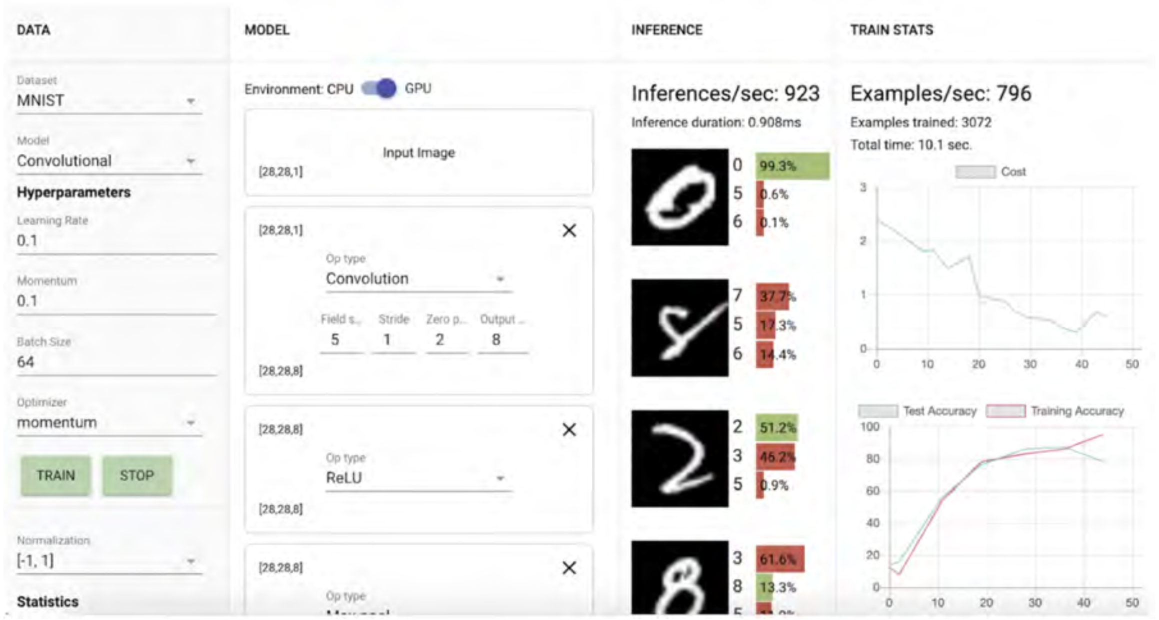


Figure 4-1: CNN Visualization Webapp from the "Use" phase of the learning progression for the CNN unit.

ML tool that allows users to drag and drop blocks of neural network layers to create simple models. They can customize these layers and train their models on various datasets, including MNIST (see Figure 4-2). The webapp also includes visualizations and graphs for training and testing. This experiential learning tool helped students gain more in-depth intuition about how model architecture (i.e. composition of layers, depth, field size) and hyperparameters (i.e. learning rate, batch size, optimizer) affect a CNN's performance on a given dataset.

In the final part of the module, students were provided a TensorFlow.js template that was organized in a very similar way to the Model Builder webapp. They were challenged with modifying parts of the code to construct and train a model that could achieve a certain accuracy on MNIST. This text-based coding experience was made much less intimidating because the content of the code file is nearly directly translatable from the previous Model Builder tool that students had become so familiar with (e.g all of the hyperparameters and layers of the neural network were organized in a



Dataset
MNIST

Model
Convolutional

Learning Rate
0.1

Momentum
0.1

Batch Size
64

Optimizer
momentum

Normalization
[-1, 1]

Environment: CPU GPU

[28,28,1] Input Image

[28,28,1]
Op type
Convolution
Field s... 5 Stride 1 Zero p... 2 Output ... 8
[28,28,8]

[28,28,8]
Op type
ReLU
[28,28,8]

[28,28,8]
Op type

Inferences/sec: 923
Inference duration: 0.908ms

0 99.3%
5 0.6%
6 0.1%

7 37.7%
5 17.3%
6 14.4%

2 51.2%
3 46.2%
5 0.9%

3 61.6%
8 13.3%

Examples/sec: 796
Examples trained: 3072
Total time: 10.1 sec.

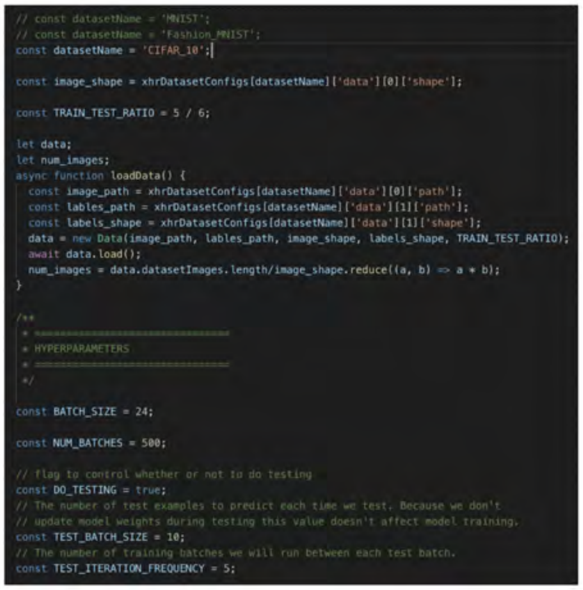
Figure 4-2: Model Builder Webapp to move students from the “Use” to “Modify” phase of the learning progression for the CNN unit [55].

similar way in the code file) (see Figure 4-3). As students work on the code template or even start a new file, they move from the “Modify” to the “Create” stage of the learning progression.

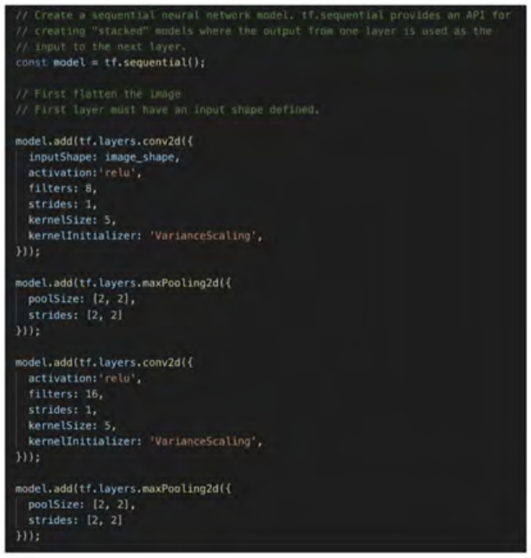
4.2.2 Self-Efficacy and Computational Action in Deep Learning Practicum

Deep Learning Practicum’s course design for teaching actionable ML emphasizes self-efficacy and incorporates several mechanisms for engagement: empowering students, creating meaningful experiences through scaffolded, inquiry-based learning, and authentic learning opportunities [138]. The four components of self-efficacy were built into the curriculum (see Figure 4-4), most notably through collaborative work with classmates, work with near-peer mentors, and exposing students to ML professionals of diverse demographics (gender, age, and race/ethnicity).

The course design of pairing students with industry mentors, inviting researchers to give guest lectures, and the final project submission as an ML blog post all served to help students develop computational identity as ML developers. By applying ML

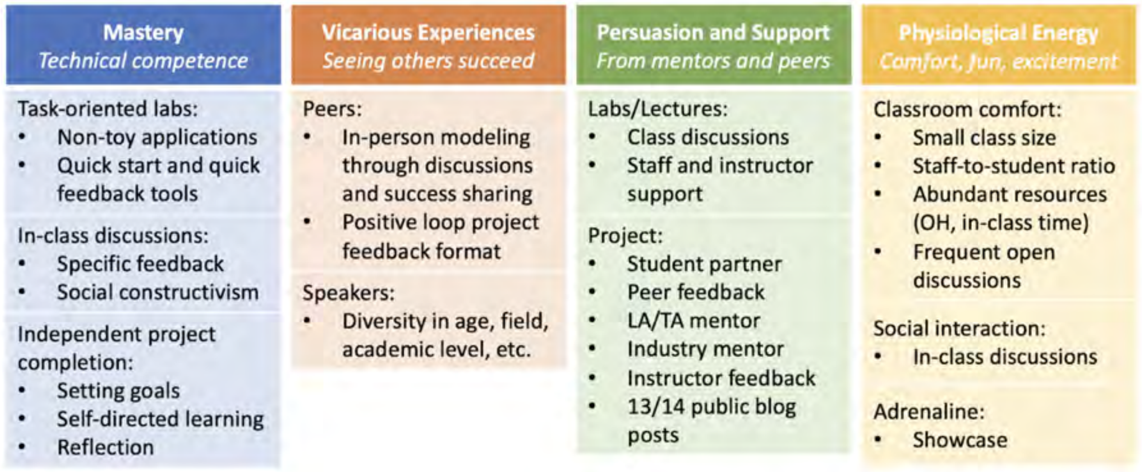


```
// const datasetName = 'MNIST';
// const datasetName = 'Fashion_MNIST';
const datasetName = 'CIFAR_10';

const image_shape = xhrDatasetConfigs[datasetName]['data'][0]['shape'];

const TRAIN_TEST_RATIO = 5 / 6;

let data;
let num_images;
async function loadData() {
  const image_path = xhrDatasetConfigs[datasetName]['data'][0]['path'];
  const lables_path = xhrDatasetConfigs[datasetName]['data'][1]['path'];
  const labels_shape = xhrDatasetConfigs[datasetName]['data'][1]['shape'];
  data = new Data(image_path, lables_path, image_shape, labels_shape, TRAIN_TEST_RATIO);
  await data.load();
  num_images = data.datasetImages.length/image_shape.reduce((a, b) => a * b);
}

/**
 * ================================
 * HYPERPARAMETERS
 * ================================
 */

const BATCH_SIZE = 24;

const NUM_BATCHES = 500;

// flag to control whether or not to do testing
const DO_TESTING = true;
// The number of test examples to predict each time we test. Because we don't
// update model weights during testing this value doesn't affect model training.
const TEST_BATCH_SIZE = 10;
// The number of training batches we will run between each test batch.
const TEST_ITERATION_FREQUENCY = 5;
```



```
// Create a sequential neural network model. tf.sequential provides an API for
// creating "stacked" models where the output from one layer is used as the
// input to the next layer.
const model = tf.sequential();

// First flatten the image
// First layer must have an input shape defined.

model.add(tf.layers.conv2d({
  inputShape: image_shape,
  activation:'relu',
  filters: 8,
  strides: 1,
  kernelSize: 5,
  kernelInitializer: 'VarianceScaling',
}));

model.add(tf.layers.maxPooling2d({
  poolSize: [2, 2],
  strides: [2, 2]
}));

model.add(tf.layers.conv2d({
  activation:'relu',
  filters: 16,
  strides: 1,
  kernelSize: 5,
  kernelInitializer: 'VarianceScaling',
}));

model.add(tf.layers.maxPooling2d({
  poolSize: [2, 2],
  strides: [2, 2]
}));
```

Figure 4-3: Coding in TensorFlow.jg to move students from the "Modify" to "Create" phase of the learning progression for the CNN unit.



Mastery *Technical competence*	**Vicarious Experiences** *Seeing others succeed*	**Persuasion and Support** *From mentors and peers*	**Physiological Energy** *Comfort, fun, excitement*
Task-oriented labs: • Non-toy applications • Quick start and quick feedback tools	Peers: • In-person modeling through discussions and success sharing • Positive loop project feedback format	Labs/Lectures: • Class discussions • Staff and instructor support	Classroom comfort: • Small class size • Staff-to-student ratio • Abundant resources (OH, in-class time) • Frequent open discussions
In-class discussions: • Specific feedback • Social constructivism	Speakers: • Diversity in age, field, academic level, etc.	Project: • Student partner • Peer feedback • LA/TA mentor • Industry mentor • Instructor feedback • 13/14 public blog posts	Social interaction: • In-class discussions
Independent project completion: • Setting goals • Self-directed learning • Reflection			Adrenaline: • Showcase

Figure 4-4: Components of self-efficacy as built into the Deep Learning Practicum course design.

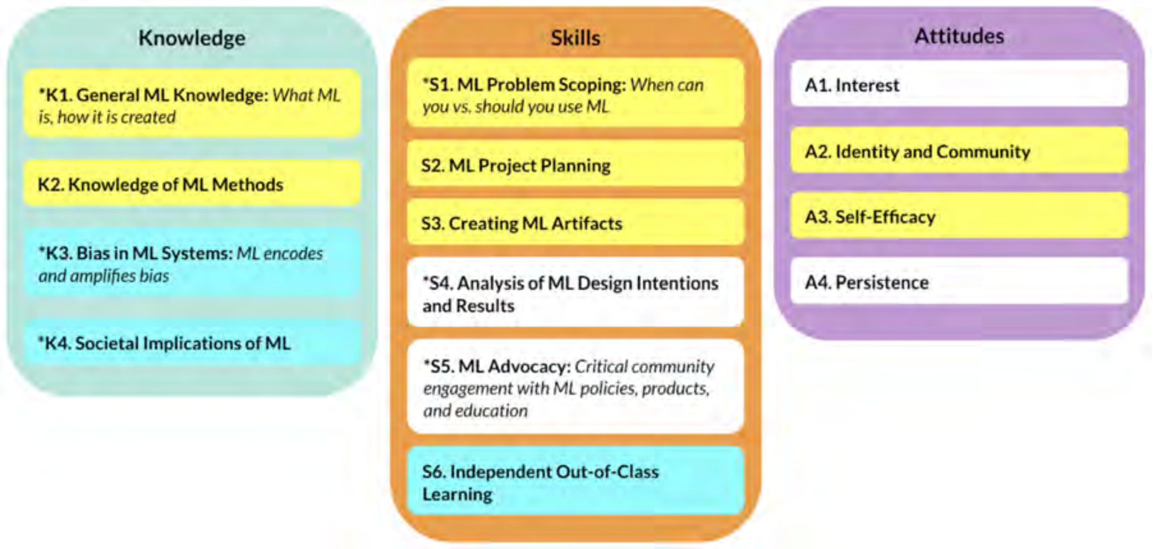

Knowledge
*K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
*K3. Bias in ML Systems: ML encodes and amplifies bias
*K4. Societal Implications of ML
Skills
*S1. ML Problem Scoping: When can you vs. should you use ML
S2. ML Project Planning
S3. Creating ML Artifacts
*S4. Analysis of ML Design Intentions and Results
*S5. ML Advocacy: Critical community engagement with ML policies, products, and education
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence

Figure 4-5: Focal (blue) and auxiliary (yellow) items from the ML Education Framework that were built into the Deep Learning Practicum curriculum.

to problems that students personally care about, the resulting projects further engage students and form a virtuous cycle capable of reinforcing and rewarding student behavior. Interwoven with approaches to support engagement and the development of self-efficacy, development of computational identity in the Deep Learning Practicum addressed how learners can connect their personal learning to community spheres.

4.2.3 Course Intentions within the ML Education Framework

The goal of Deep Learning Practicum is to be a *technical* course that aims to develop students into ML *tinkerers* who are able to hack at and build useful ML applications for themselves, instead of being a technical course that tries to help students become ML researchers or engineers who need to understand the detailed mathematics behind deep learning. In accordance with these goals, we built 7 focal items from the ML Education framework into the curriculum and also included 3 auxiliary items that were lightly covered during the course (see Figure 4-5).

There were 2 focal items in Knowledge, 3 in Skills, and 2 in Attitudes that we heavily built into Deep Learning Practicum. Under Attitudes, these are A3. Self-Efficacy and A2. Identity and Community. Self-efficacy in ML was the primary goal of the courses and was built through hands-on, technical modules that employed the Use-Modify-Create progression and through the final capstone project assignment. Students developed a sense of identity as ML tinkerers through in-class discussions about the ML artifacts they created throughout the course with a learning community of their peers, as well as through interactions with mentors from industry.

Under Skills, Deep Learning Practicum focused on S1. through S3. for Problem Scoping, Project Planning, and ML application creation. S1. and S2. were developed through several assignments and activities where students attempted to scope out several project ideas for ML with feedback from their peers and the course staff. For the final project, they also submitted a detailed project proposal as well as proposals for intermediary processes in creating ML applications (i.e. data collection, model training).

Under Knowledge, Deep Learning Practicum hoped to teach K1. General ML Knowledge and K2. Knowledge of ML Methods. Students gained clear understandings of what ML is and the pipeline of how ML applications are created through hands-on activities in building ML systems throughout the course. They learned a range of deep learning and transfer learning methods for K2.

The auxiliary items of K3. Bias in ML Systems and K4. Societal Implications of ML were included through several guest lectures by research and industry leaders in these fields. Students were also sometimes assigned small portions of assignments from the technical modules relating to these topics. The final auxiliary item, S6. Independent Out-of-Class Learning, was indirectly built into the final capstone project portion of the course since students almost always had to do extensive research on datasets or ML methods for their capstone projects.

We hoped that the in-depth learning experiences from the focal and auxiliary items would indirectly help students develop some ability in the remaining items.

4.3 Transfer Learning for Empowering Beginners in ML Application Creation

The ML method of transfer learning was prominently featured in the Deep Learning Practicum curriculum and was at the core of how students with minimal programming and ML knowledge were able to successfully create personally meaningful and useful ML applications after only one semester. Transfer learning involves using an existing machine learning model to train a new model specialized for a slightly different task [99]. Although the design and testing of complicated model architectures is inaccessible to most people, repurposing the trained model is accessible to developers of all levels and students of any age. Starting with a powerful pre-trained model allows new developers to apply deep learning to solve novel problems without the vast computational resources and time needed to design model architectures and train and test neural networks from scratch, which can take thousands of dollars and months of time. Transfer learning has allowed machine learning enthusiasts to build their own models using complex models as a starting point. In Deep Learning Practicum, transfer learning was a technique taught from the first module. Even as students built their own models from scratch in later units to develop their knowledge of specific ML methods, most students used a combination of transfer learning techniques and architecting their own smaller models in their capstone projects, which allowed them to quickly create powerful, yet still personalized applications.

4.4 Deep Learning Practicum Version 1

The goal of the pilot version of the course was to engage students in a deeper understanding of the capabilities and limitations of ML through guided investigations and open-ended experimentation, and to prepare students to develop capstone projects. The course did not count towards core undergraduate requirements and was an elective course. In pre-registration, the instructors emphasized that the course was meant for students who did not already feel comfortable working with machine learning and not experts hoping to gain more advanced techniques. Participating students registered for the course because of a personal interest in the topic and a desire to get hands-on experience in ML. The class size was restricted due to the intensive, personalized, and project based nature of instruction. Twelve students completed the course.

During the course, the instructors aimed to ground theoretical constructs of ML in applications that spanned different topics. The 6 genres covered in the pilot included both predictive and generative applications of ML. The instructors would give short (not exceeding 15 minutes) explanatory technical lectures and provide in-class exercises in TensorFlow.js for students to try on their own laptops. The activities often leveraged existing datasets, pre-built models, and web-based tools for ML. During or after each set of exercises, students were asked to discuss their findings with a partner or with the whole class. Teaching assistants (TAs) and lab assistants (LAs) provided technical and instructional support to the instructors during the exercises. There were weekly take-home assignments that provided an extension to the environment and the exercises introduced during class. The progression of genres followed the historical development of ML, and thus presented a progression in the sophistication of ML models themselves. There were 3 starter topics followed by 3 advanced topics.

The last 9 weeks of class were spent on capstone projects and guest lectures. Students were asked to find a problem that personally interested them and was suitable for a machine learning solution. Course staff and industry mentors were paired to each project. Guest lectures from leaders in research and industry were interspersed. A week-by-week map of the version 1 curriculum is presented in Table 4.1.

4.4.1 Student Demographics

The 12 students were diverse in academic level, major, and background. Two (17%) were second-year students, 4 (33%) were third-years, 5 (42%) were fourth-years, and 1 (8%) was a graduate student. Nine students (75%) majored in EE/CS, 1 majored in Math, 1 in Math and Physics, and 1 in Humanities. There were 3 Black women, 3

Week	Topics	Progression
1	K-nearest neighbors	**Use:** Teachable Machine webapp [118]. **Modify:** Confidence algorithms in source code.
2	Multilayer Networks	**Use:** Model Builder webapp [55]. **Modify:** Starter TensorFlow.js and HTML code for programming multilayer networks.
3	CNNs	**Use:** Model Builder webapp, filter visualization webapp [44], Fast Style Transfer webapp [88]. **Modify:** Starter code for programming CNNs.
4	Generative Models and Embeddings	**Use:** Embedding Projector webapp [125], Latent Space Explorer [22]. **Modify:** Feature projection functions in Latent Space Explorer source code.
5	GANs	**Use:** GAN Playground webapp [87]. **Modify:** Starter TensorFlow.js and HTML code for programming GANs.
6	RNNs and LSTMs	**Use:** Andrej Karpathy's RNN text generation webapp [64], SketchRNN webapp [43]. **Modify:** Architecture and parameters in webapp source code. **Create:** Music generation RNN application through TensorFlow Python notebook.
7	Project Overview	**Create:** Students scoped and presented 3 project ideas.
8	Spring Break	**Create:** Teams/individuals worked on the project proposal writeup.
9	Project Mentor Matching, Guest Lecture: ML in Healthcare	**Create:** Teams submitted proposals and were matched with industry mentors.
10	Guest Lecture: Fairness, Guest Lecture: Testing and Training Tools	**Create:** Teams work on projects.
11	Guest Lecture: Interpretability	**Create:** Teams work on projects.
12	Project Midpoint Checkpoint Presentations, Guest Lecture: Music and Art	**Create:** Teams presented 5-minute project progress reports in class, received feedback.
13	Guest Lecture: People and AI Research	**Create:** Work on projects.
14	Final Presentation Dress Rehearsal, Guest Lecture: Adversarial Examples	**Create:** Teams presented a practice run of their final 10-minute project presentations in class, received feedback.
15	Final Presentation Showcase, Project Writeup Due	**Create:** Present projects in front of industry professionals and submit project writeups in the form of instructional blog posts.

Table 4.1: Spring 2018 week-by-week curriculum annotated with the ITEST Use-Modify-Create progression.

Asian men, 2 Asian women, 2 White or Caucassian women, and 1 White or Caucassian man. Ten students (83%) had taken an introductory or online course in AI or ML, and commented in the pre- survey that they wanted to take this introductory course because they did not feel that they had the ability to build practical applications. Two students (17%) had no course experience in AI or ML. Four students (33%) had no experience in JavaScript, 2 (17%) had some experience, and 6 (12%) had significant experience.

4.4.2 Teaching Staff and Industry Mentors

There were 6 TAs/LAs with no differentiation in the roles. The staff were instrumental in debugging in-class exercises for each topic, answering questions, and leading reflective discussions that helped students achieve learning goals for the exercises. For the 9 projects in the class, 3 of the TAs/LAs mentored 1 project each and 3 mentored 2 projects each. There were 9 industry mentors. We reached out to companies and researchers in the area to ask for volunteers who have experience with ML projects. We invited all volunteers to a mixer with the students after project teams had formed. At the beginning of the mixer, each mentor gave a brief overview of their expertise and each student team summarized their project goals. After the mixer, student teams submitted their preferences for mentors and were matched. Mentors mostly met with teams during the beginning and middle of their project cycles to help with higher level ideas, resources, and project scoping.

4.4.3 Capstone Projects

Within this "Create" stage of the course, students marshaled the tools and techniques at their disposal along with mentor feedback to create capstone projects. Instructors emphasized that students should choose a project that they were personally interested in, but also cautioned that "a realistic project implemented well and evaluated thoroughly is better than a half-implemented ambitious project with no result." Projects could either be a real-world "**Application**" of ML, an "**Exploration**" of properties of neural networks, or a "**Replication**" of an ML paper. To help scaffold project scoping, students were given a "3 Ideas" assignment in which they were required to present 3 project ideas to the class. For each idea, students defined a "Safe" goal that they were confident they could achieve by the end of the semester, a "Target" goal that they hoped to achieve, and a "Stretch" goal that would be good to achieve if extra time was available. Students were given time after the presentations to talk with others in the class and optionally find a partner.

There were 9 total project groups consisting of 3 pair projects and 6 solo projects. 7 projects were in the "Application" category, 1 in "Replication," and 1 in both "Application" and "Exploration." All teams achieved the "Safe" goal for their project. One team continued working on their project with their industry mentor after the class and was able to publish a paper.

4.4.4 Learnings from Version 1 that Informed Version 2

Feedback was obtained through the post- survey and a discussion-style post-mortem on the last day of class. Due to the small class size, quantitative analysis of weekly assignment grades and survey responses are not presented. Overall, students loved the interactive lab style of the modules portion of the class. Two students with no prior JavaScript experience felt that the course was surprisingly language-independent and that JavaScript knowledge was not required, although some coding experience would be helpful. Students felt that the small class size was beneficial in creating a relationship between students and staff that made them feel comfortable speaking up during the open reflective discussions that accompanied in-class exercises. Nearly every student felt that there was not enough time for project implementation, but students also said that it was the most valuable and enjoyable part of the course. Students suggested that the modules portion of the class should cover data collection, web scraping, data processing, and connecting to computational resources to better scaffold the projects. Students enjoyed the guest lectures and thought that they helped "put what we learned into a much bigger picture." Students noted that some guest lectures may have been helpful before starting their final projects and would have provided additional context for final project choices.

Several students said that the course demystified the field of machine learning and made it more approachable. Two students mentioned their increased concern over bias in machine learning algorithms as well as a deeper understanding of how to resolve some of these issues. One wrote: "*Before this course, I thought of computer programs more linearly – as if computer programmers were mostly in control of a program's results. Now I have a much greater understanding of how machine learning programs can be biased and unfair... I learned the importance of providing good, varied input data and how this data can have a significant impact on a network and ultimately the world.*"

4.5 Deep Learning Practicum Version 2

4.5.1 Changes from Version 1

The second version of Deep Learning Practicum was offered at Massachusetts Institute of Technology in fall 2018, the semester following the pilot. There were 6 main changes from version 1: (1) the final project was introduced at the beginning of the semester and ran in parallel to the modules portion of the course, (2) there were two additional scaffolding workshops for the final project (data mining and using computing clusters), (3) students were required to have a partner for their project unless given permission, (4) guest lectures were more interspersed throughout the course instead of all at the end, (5) the staff-to-student ratio decreased from 6:12 to 6:26, and (6) an additional unit on reinforcement learning was added. The full set of version 2 curricula, lectures, assignments, and final projects can be found online at `mit.edu/6.s198` [70]. A week-by-week map of the version 2 curriculum is presented in Table 4.2.

Week	Topics	Progression
1	K-nearest neighbors, Transfer learning	Module same as version 1 week 1 with more emphasis on transfer learning techniques.
2	Multilayer Networks, projects overview	Module same as version 1 week 2. **Create:** Scope 3 ideas for capstone final project.
3	CNNs, guest lecture	Module same as version 1 week 3. **Modify:** Starter adversarial example TensorFlow code from guest lecture.
4	3 Ideas project workshop, Data Mining workshop	**Use:** Kaggle to find datasets [54]. **Modify:** 3 project ideas based on feedback. **Create:** Web scraping scripts using Beautiful Soup [86].
5	Generative Models and Embeddings, Computational Resources workshop, Project mentor matching	Module same as version 1 week 4. **Use:** The Massachusetts Green High Performance Computing Center tutorial [80].
6	GANs	Module same as version 1 week 5.
7	Project Data Review, Reinforcement Learning	**Use:** Metacar webapp [127], OpenAI Gym webapps [98]. **Modify:** TensorFlow starter code for RL. **Create:** Data review document to describe project dataset details.
8	RNNs and LSTMs	Module same as version 1 week 6.
9	Informal project checkpoint	**Create:** Work on projects and discuss progress with staff.
10	Guest lectures	**Create:** Work on project proposal.
11	Formal project checkpoint	**Create:** Work on projects and show basic working demo to staff.
12	Guest lecture, Project practice lightning talks	**Create:** Present 2-minute project lightning talks, receive feedback for final showcase.
13	Office hours	**Create:** Work on projects.
14	Final Presentation Showcase	**Create:** Presented capstone projects to an audience of varying ML experience with lightning talks, then individual booths.
15	Project Writeup Due	**Create:** Submit project writeups in the form of instructional blog posts.

Table 4.2: Fall 2018 week-by-week curriculum annotated with ITEST's Use-Modify-Create progression.

4.5.2 Student Demographics

Of the 26 students, there were 5 (19%) second-years, 6 (23%) third-years, 11 (42%) fourth-years, 3 (12%) graduate students, and 1 (4%) post-doc. Twenty students (77%) majored in EE and/or CS, 2 (8%) in Architecture, 2 (8%) in Physics and EECS, 1 (4%) in Materials Science and Engineering, and 1 (4%) in Biological Engineering and Math. There were 9 (35%) Asian women, 6 (23%) Asian men, 5 (19%) White or Caucassian men, 3 (12%) Hispanic men, 1 (4%) Black woman, 1 (4%) White or Caucassian woman, and 1 (4%) Black man. Similar to the previous offering, 21 students (81%) had taken an introductory course in AI or ML, but commented that they wanted to participate in the course due to self-perceived lack of ability to apply theory and math in building practical applications.

4.5.3 Capstone Projects

For the second offering, students were required to find a partner for the final project unless otherwise approved due to the decrease in the staff-student ratio and the successes from the pilot. There were 14 projects, all of which completed their "Safe" goal. All three project categories were represented with the majority being "Application" projects. More projects bridged multiple project categories than in the first iteration, likely due to students having more time. Students started to work on final projects in week 2 of the course, so they had not been exposed to all of the topic modules. The instructors were concerned that students may avoid later topics and tried to mediate this by giving lightning talks and sample use cases for the topics that would be presented later.

The first project workshop was the 3 Ideas Workshop during week 4, which changed in format from the pilot due to the increased number of students. The staff ran two 30-minute sessions of guided group presentations. For each session, the class was divided into four groups of 5-6 students based on shared project topic interests. One to two TAs/LAs led each group, where students took turns presenting their 3 ideas. During the final 30 minutes of the class, students were encouraged to talk to others they had met and form groups. After the 3 Ideas Workshop, there were two workshops given on project skills: data collection and how to connect to computing resources. There was also a data review checkpoint assignment due in week 7 to confirm that students had completed data collection and processing in a timely manner.

4.6 Findings and Analysis

Responses to the optional post-course survey and all of the student capstone projects informed the analysis of the course. The post- survey was emailed out after the end of the course and received 17 responses (65%). Survey demographic results indicated that the students who took the survey were fairly representative of the class in terms of diversity and experience. All grade levels and majors of course-takers were represented. There were 1 sophomore, 5 juniors, 9 seniors, and 2 graduate students. There were 12 students majoring in EE/CS, 2 in Physics and EECS, 1 in Materials Science and Engineering, 1 in Architecture, and 1 in Mathematics and Biological Engineering. 12 (71%) students who took the survey were men and 5 (29%) were women. 10 were Asian (59%), 3 were White or Caucasian (18%), 2 were Hispanic (12%), and 2 were Black (12%). There were 14 completed projects in the class. One student was given permission to do a solo project based on the scope of the work. Another student had a partner who dropped the class.

I will discuss the results in five sections: as pertaining to projects, development of self-efficacy (A3), knowledge of ML methods (K2), ability to perform skills relating to ML application creation (S1-S3), and Identity and Perceptions (A1, A2).

4.6.1 Project Themes and Approaches

The end-of-course survey asked respondents to indicate how they chose the topic for their capstone project. Twelve students (71%) indicated that they chose their topic based on technical interest, 8 (47%) on a personal hobby, 4 (24%) on perceived usefulness in career/research, 3 (18%) on social good, and 3 (18%) followed their project partner's interests. The end-of-course survey also provided evidence of mastery over ML concepts and skills through creating final projects. When asked "Which of the following elements from the course did you use in your project work?", the responses were as follows:

- "I used concepts/architectures from the units": 14 (82%)
- "I used independent researching skills that I developed through the assignments": 10 (59%)
- "I used the 3 Ideas Workshop to help me improve or refine my project idea": 10 (59%)
- "I used techniques from the Data Gathering workshop": 7 (41%)
- "I used coding skills that I developed through the assignments": 6 (35%)
- "I used debugging skills that I developed through the assignments": 6 (35%)
- "I used the 3 Ideas Workshop to help me find a partner": 6 (35%)
- "I used code directly from class assignments": 1 (6%)

Three of the 14 projects are detailed below to show the variety of project categories, topics, and methods used.

Project 1: Generating Virtual Worlds

This was a pair **Application** project by one non-EECS student and one EECS with the goal of creating new video game environments. The students were motivated by a recent video game that had algorithmically generated 18 quintillion planets for players to explore and expressed amazement that "infinite unique possibilities can be created with [relatively little] work from humans" through procedural generation.

The project used an ensemble method. They trained a Generative Adversarial Network on terrain heightmaps to create the base height maps for the new world, and then passed the results through a fast style transfer program trained on Earth satellite images to obtain a color mask for the base terrain.

The team identified websites that provided free terrain heightmap images and color satellite images. They did not have experience with web scraping and received external scripting help to avoid manually downloading the images. They collected 10,000 black-and-white heightmaps and 56,000 satellite images.

The students designed 13 different versions of their GAN for generating the base terrain heightmaps, with modifications to architecture and hyperparameters. They scaled the training images down to 256x256 pixels to decrease training time. To add color and texture to these black-and-white height maps, the team used style transfer, which refers to the concept of taking a content image and a style image and combining the features to produce an output image with the content of one image and the style of another [38]. The team started by feeding an open source Fast Style Transfer model 512x512 pixel color satellite images [30], but found that they were quickly running out of memory before the program could begin training. To address storage and time constraints, the team modified the hyperparameters of the model, rescaled the training images to 128x128, and decreased the number of images from 56,000 to 28,000. They also debugged a severe artefact issue by replacing transpose convolution layers with bilinear image upscaling, which significantly modified the architecture of the original Fast Style Transfer model.

For the final demo, the team was able to generate colorful worlds that they imported into a gaming environment so that a player could drive around and explore the terrain. They also experimented with how their model translated into the physical world via 3D printing (see Figure 4-6)

Through analysis of the team's project write-ups, presentation, and demo, Figure 4-7 shows how the project engages with various aspects of the ML Framework. It demonstrated high engagement with the 2 Knowledge items, 4 Skills, and 2 Attitudinal items highlighted in yellow and some engagement with the 1 Skill and 1 Attitudinal item highlighted in blue. The project was rather technical and implementation-focused, which required high engagement with K1. General ML Knowledge, K2.

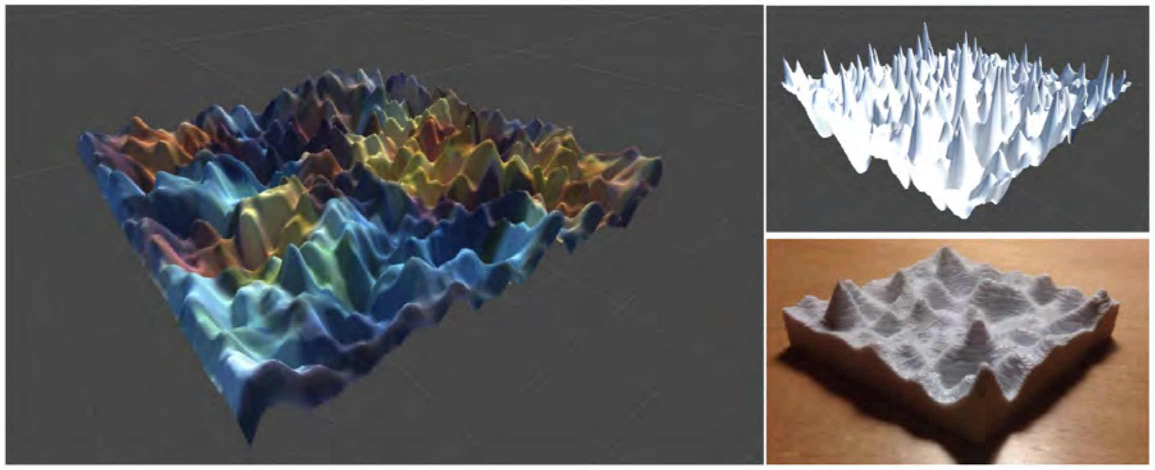


Figure 4-6: "Generating Virtual Worlds": Final game environment outputs displayed in Unity3D. Left: Colored heightmap 1, Right-top: Uncolored heightmap 2, Right-bottom: 3D printed heightmap 3.

Knowledge of ML Methods, and S1.-S3. Skills associated with creating ML applications. The students also did significant research into training and modifying GANs and the Fast Style Transfer model outside of class (S6. Independent Out-of-Class Learning). The selection of the project topic and successful implementation indicated high levels of A1. Interest and A3. Self-Efficacy.

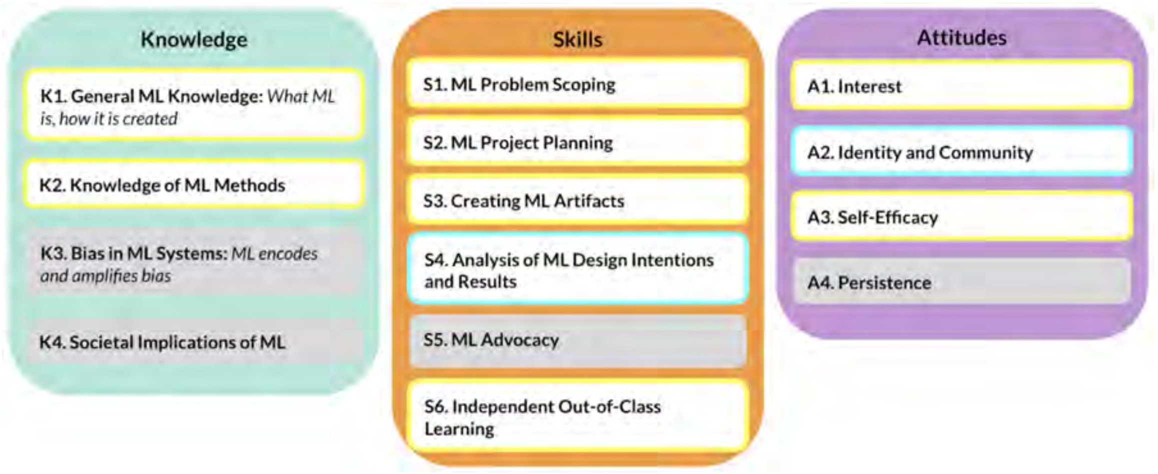

Knowledge
K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
K3. Bias in ML Systems: ML encodes and amplifies bias
K4. Societal Implications of ML
Skills
S1. ML Problem Scoping
S2. ML Project Planning
S3. Creating ML Artifacts
S4. Analysis of ML Design Intentions and Results
S5. ML Advocacy
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence

Figure 4-7: Analysis of engagement with the ML Education Framework in student project "Generating Virtual Worlds". Items of high engagement are in yellow, items with some engagement are in blue, and all other items are in gray.

The following concepts (sub-items under K2. Knowledge of ML Methods) from the modules portion of the class were heavily used in this project:

- **Multilayer Networks**: Linear and activation layers were used to build the GAN model and to modify the Fast Style Transfer model.

- **Convolutional Neural Networks**: CNN layers were used to build the GAN model and in the Fast Style Transfer model.

- **Generative Adversarial Networks**: A GAN was used to generate new images.

The following skills introduced through exercises and homework assignments from the modules portion of the class were heavily used in this project, as indicated by the student's reflection in their writeup:

- **Choosing Models**(sub-skill of S2. ML Project Planning): The team found Fast Style Transfer suitable for the problem at hand and used it as a core piece of the latter half of their project.

- **Modifying Models** (sub-skill of S3. Creating ML Artifacts): The team significantly modified the architecture of the Fast Style Transfer model to debug artefacts that arose.

- **Modifying Learning Rates**, **Modifying Layer Hyperparameters**, and **Modifying Batch Sizes** (sub-skills of S3. Creating ML Artifacts): "...tweaks include changing the discriminator and generator's learning rates, number of layers, type of layers, convolutional filter sizes, and image batch size."

- **Creating Models** (sub-skill of S3. Creating ML Artifacts): The team created their own GAN model.

- **Creating Loss Functions** (sub-skill of S3. Creating ML Artifacts): "Getting the distance between [the new style image's gram matrix] and the original style gram matrix gives us the style loss. A similar procedure can be done to find content loss. The model calculates the total loss by summing these two values."

- **Training and Testing** (sub-skill of S3. Creating ML Artifacts, and S4. Analysis of ML Design Intentions and Results): Both models trained over several epochs. Testing was objective in terms of how “cool” the output worlds looked from the perspective of the creators.

Project 2: Analyzing Racial Bias among Politicians’ Tweets on Mass Shootings

This was a solo **Exploration** project by a non-EECS student with the goal of answering the questions: "How do elected politicians characterize perpetrators of mass shootings differently?" and "Does this correlate with established racial stereotypes?" The student obtained a dataset by sorting through a database of tweets authored by members of the 115th Congress, filtered by hashtags for recent mass shootings and keywords such as "shooting", "attacker", and "gun". The student experimented with training a word embedding model from scratch and using transfer learning on the Google word2vec model trained on the Google News Corpus [81]. After training the final model using transfer learning, the student used methods from the Word Embedding Association Test (WEAT) to analyze the relationship between certain bias-indicative terms and the political party of the author [62]. WEAT was an algorithm that she researched and discovered outside of class to apply to this project.

The student found correlations such as: "Democrats associated sadness with the shooter where Republicans associated anger...Democrats characterize shooters more often as being alone where Republicans characterize them as part of a group...there is a nearly but not quite significant suggestion that Democrats associate shooters with the thug stereotype slightly more than Republicans." The student did not have enough time to compare results based on the shooter’s race, but outlined how their model and process would be modified in future works (see Figure 4-8).

Through analysis of the student’s project write-ups, presentation, and demo, Figure 4-9 shows how the project engages with various aspects of the ML Framework. It demonstrated high engagement with the 3 Knowledge items, 4 Skills, and 2 Attitudinal items highlighted in yellow and some engagement with the 2 Skills and 1

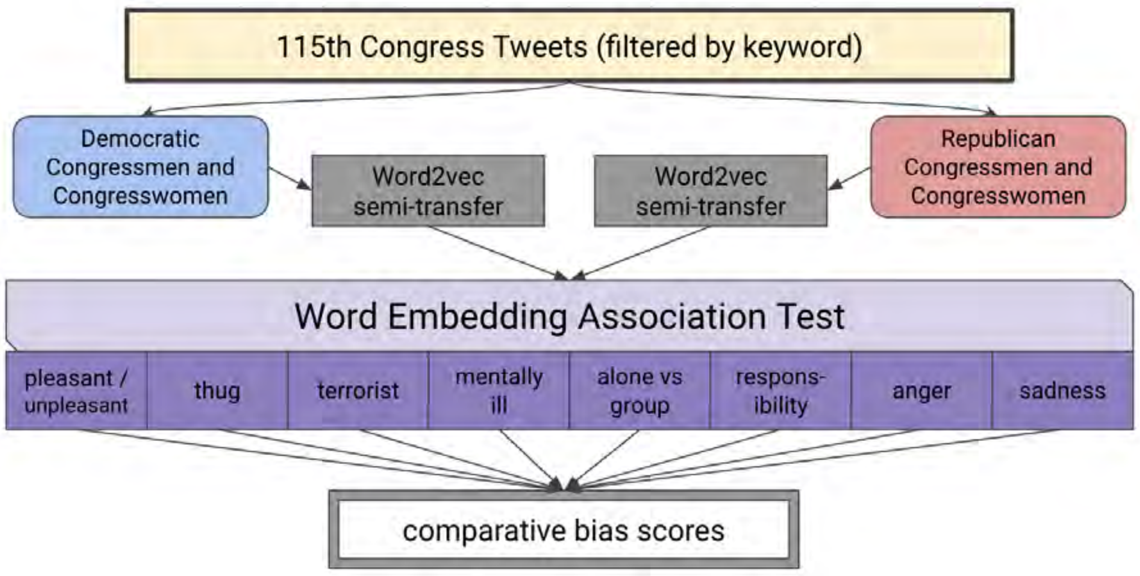

115th Congress Tweets (filtered by keyword)
Democratic Congressmen and Congresswomen
Word2vec semi-transfer
Word2vec semi-transfer
Republican Congressmen and Congresswomen
Word Embedding Association Test
pleasant / unpleasant
thug
terrorist
mentally ill
alone vs group
respons-ibility
anger
sadness
comparative bias scores

Figure 4-8: "Analyzing Racial Bias among Politicians' Tweets on Mass Shootings": Model and process map.

Attitudinal item highlighted in blue. This project was not very focused on creating ML applications. Instead, the student was more involved in applying different ML tools to analyze a set of data that addressed a problem that she was very passionate about. This led to high engagement with S5. ML Advocacy, K4. Societal Implications of ML, and S6. Independent Out-of-Class Learning. She also expressed high levels of A1. Interest in ML through her exploration of her chosen topic and also had high A3. Self-Efficacy in ML.

The following concepts (sub-items under K2. Knowledge of ML Methods) from the modules portion of the class were heavily used in this project:

- **Transfer Learning**: The student used transfer learning on word2vec to train the final model.
- **Multilayer Networks**: The student built a simple neural network on top of word2vec for transfer learning.
- **Embeddings and Generative Models**: The student used word embeddings and WEAT for her analysis.

The following skills introduced through exercises and homework assignments from

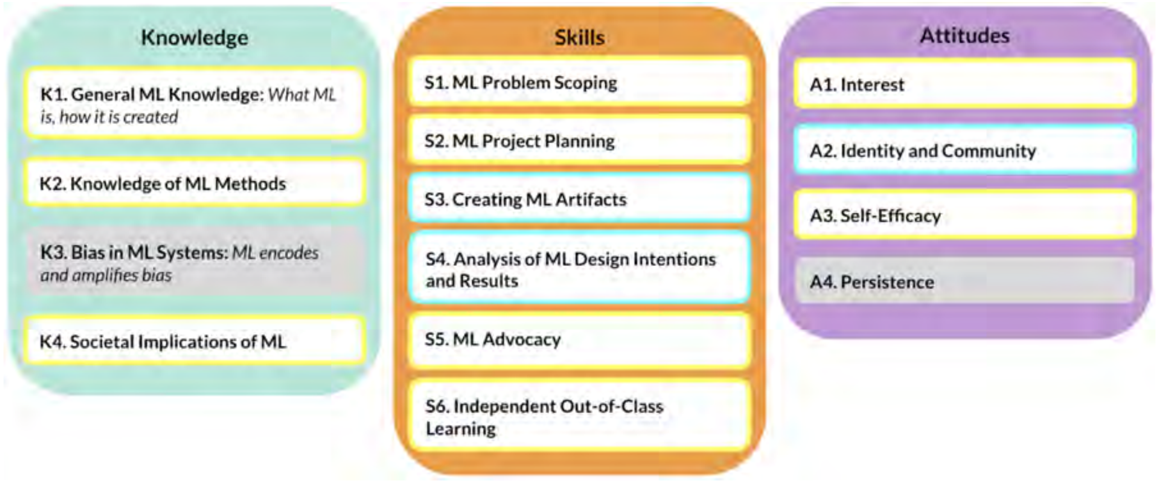

Knowledge
K1. General ML Knowledge: What ML is, how it is created
K2. Knowledge of ML Methods
K3. Bias in ML Systems: ML encodes and amplifies bias
K4. Societal Implications of ML
Skills
S1. ML Problem Scoping
S2. ML Project Planning
S3. Creating ML Artifacts
S4. Analysis of ML Design Intentions and Results
S5. ML Advocacy
S6. Independent Out-of-Class Learning
Attitudes
A1. Interest
A2. Identity and Community
A3. Self-Efficacy
A4. Persistence

Figure 4-9: Analysis of engagement with the ML Education Framework in student project "Analyzing Racial Bias among Politicians' Tweets on Mass Shootings". Items of high engagement are in yellow, items with some engagement are in blue, and all other items are in gray.

the modules portion of the class were heavily used in this project:

- **Choosing Datasets** (sub-skill of S2. ML Project Planning): The student found a suitable pre-compiled dataset of tweets to work from.
- **Creating Datasets** (sub-skill of S3. Creating ML Artifacts): The student algorithmically filtered through a dataset to create a custom dataset for the project.
- **Choosing Models** (sub-skill of S2. ML Project Planning): The word2vec model trained on the Google News corpus was selected for transfer learning.
- **Using Models** (sub-skill of S3. Creating ML Artifacts): The student used the results of the word embeddings model in a critical way to calculate effect values and p-values for relevant terms.
- **Training and Testing** (sub-skill of S3. Creating ML Artifacts, and S4. Analysis of ML Design Intentions and Results): Analogy completion was selected as an external standardized metric to evaluate the performance of trained word embeddings models.

Project 3: Adversarial Infill

This was a team **Replication** project by two EECS students with the goal of exploring local and non-local methods for semantic infilling. Specifically, the students wanted to try to re-implement and improve on the GAN infilling method using TensorFlow [139]. The team used the publicly available CelebA dataset that was also used in the original paper, which contains 202,599 images of celebrity faces [75]. The students found that "the biggest issue with this infilling method is that the boundary region is obvious to the naked eye", and in order to improve upon this issue, they designed their own loss function that included a boundary blending loss, "which minimizes pixel differences around rectangular masked borders". The team did research on various GAN implementations and successfully built and trained a GAN that had 4 convolutional layers in both the generator and the discriminator. They tested center mask, random mask, and pattern mask on their model using the CelebA dataset and were able to achieve results that were smoother than prior methods. For the final showcase, the team took pictures of audience members, ran a center mask, and had their model infill the missing piece as a live demo (see Figure 4-10).

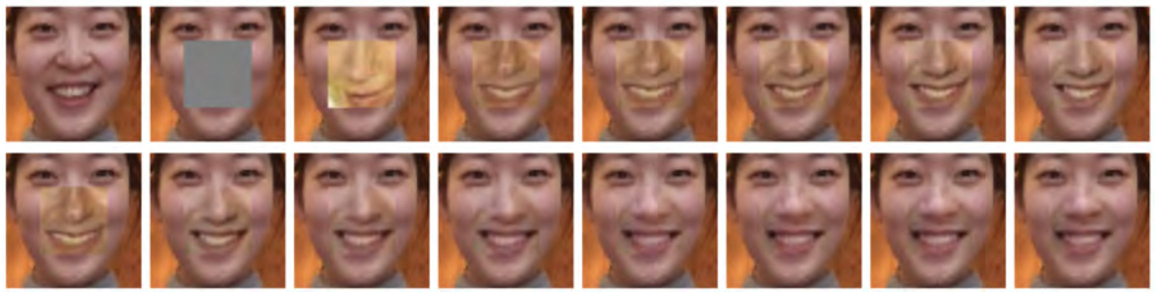


Figure 4-10: "Adversarial Infill": Example of the model running on a laptop webcam photo over several iterations.

This project engaged with the ML Framework in similar levels to the "Generating Virtual Worlds" Application project (see Figure 4-7). The project was ideated as an extremely technical research project that the students took as a fun challenge, and it required high engagement with K1. General ML Knowledge, K2. Knowledge of ML Methods, and S1.-S3. Skills associated with creating ML applications. The students also did significant research into designing loss functions and creating GANs outside of class (S6. Independent Out-of-Class Learning). The selection of the project topic and successful implementation indicated high levels of A1. Interest and A3. Self-Efficacy.

The following concepts (sub-items under K2. Knowledge of ML Methods) from the modules portion of the class were heavily used in this project:

- **Convolutional Neural Networks**: The layers of the generator and discriminator were made up of 2D CNNs.
- **Generative Adversarial Networks**: A GAN was used for adversarial infilling.

The following skills introduced through exercises and homework assignments from the modules portion of the class were heavily used in this project:

- **Choosing Datasets** (sub-skill of S2. ML Project Planning): The team found and used the public CelebA dataset.
- **Choosing Models** (sub-skill of S2. ML Project Planning): The team chose to replicate the GAN infilling paper after looking at several other techniques.
- **Creating Models** (sub-skill of S3. Creating ML Artifacts): The team coded their own GAN model based on literature: "On account of the fact that adversarial training is delicate, and all GANs are variations of Ian Goodfellow's original GAN, it was useful to use an existing GAN implementation."
- **Modifying Loss Functions** (sub-skill of S3. Creating ML Artifacts): The team learned from literature to "update the generator's loss twice for every discriminator update; this is based on the diminished gradient problem where the discriminator loss converges to zero."
- **Creating Loss Functions** (sub-skill of S3. Creating ML Artifacts): The team created a 3-part loss function to account for context, realistic perception, and boundary blending.
- **Modifying Learning Rates** and **Modifying Batch Sizes** (sub-skills of S3. Creating ML Artifacts): Experimentally, the team found that "the best learning

	Item	μ	σ
1	I felt that I was successful in this class.	4.4	0.6
2	I am proud of what I was able to accomplish in my final project.	4.2	0.8
3	I will be able to complete a ML project (of a similar level and scale to my final project) on my own.	4.6	0.5
4	In this class, I saw people like me succeed at learning ML.	4.2	0.6
5	When I saw people like me succeed in ML, it made me feel that I could succeed as well.	4.3	0.7
6	How confident do you feel about describing your project to a non-technical person?	4.6	0.5
7	The project work made me feel uncomfortable	1.6	0.9

Table 4.3: Means and standard deviations of post- survey linear scale question responses.

rate for [the] Adam optimizer was 0.0001. Also, a batch size of 32 worked better than 64."

- **Training and Testing** (sub-skill of S3. Creating ML Artifacts, and S4. Analysis of ML Design Intentions and Results): The team conducted many experiments and found that "overfitting usually started to occur after 30 epochs" and tested the results of their models subjectively and by looking at the results of loss functions.

4.6.2 Self-Efficacy

In this subsection we describe student outcomes associated with each of the four components of self-efficacy: mastery experiences, vicarious experiences, persuasion/supportive messages, and physiological reactions. See Table 4.3 for a summary of all responses to the linear scale questions on the post-course survey, where 5 = “Strongly agree” or “Completely confident” and 1 = “Strongly disagree” or “No confidence.”

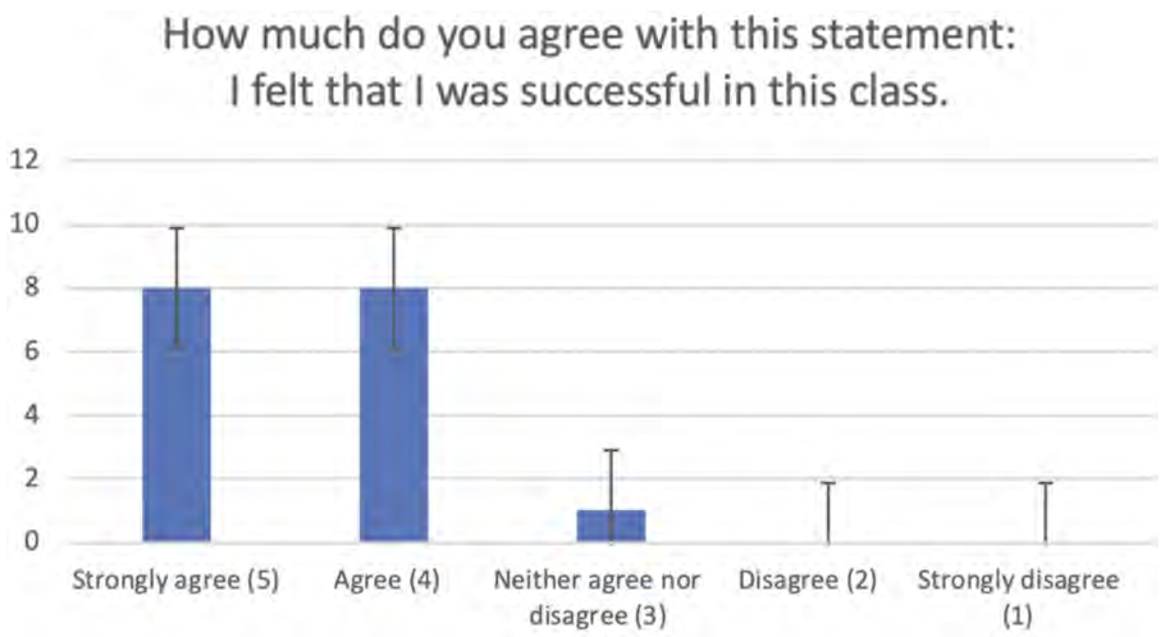
Mastery experiences

Mastery experiences are task-specific experiences of success and are the primary contributor to one's self-efficacy beliefs. Students with high self-efficacy believe they are able to complete tasks similar or closely related to the task that served as the basis for the mastery experience. The instructors sought to raise students' self-efficacy through "mastery experiences" that increased their knowledge and skills. A set of free response, checkbox, and linear scale questions on the end-of-course survey captured students' self-efficacy in ML.

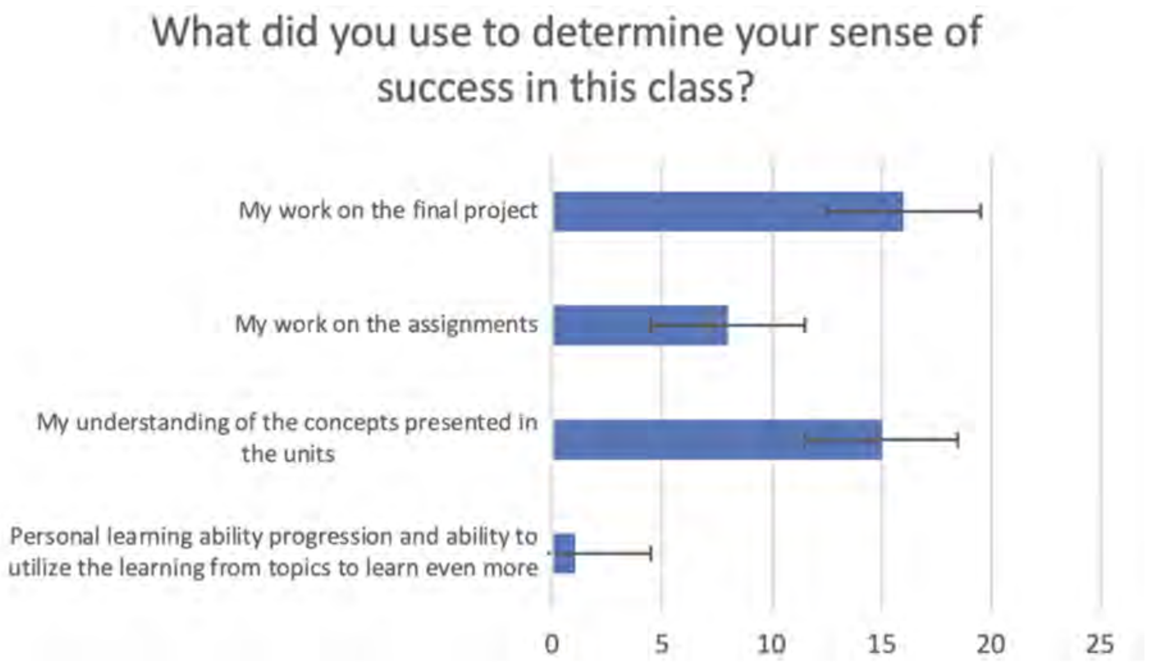
An item that was particularly relevant to self-efficacy was Question 1 of Table 4.3, a Likert scale question for the statement: "I felt that I was successful in this class." Of the respondents, 47% strongly agreed, 47% agreed, 6% neither agree nor disagree, and 0% disagree or strongly disagreed (see Figure 4-11a). As a follow-up to Question 1, students were asked "What did you use to determine your sense of success in the class?" Respondents attributed their sense of success most with work on the final project (94%), which is most associated with A3. Self-Efficacy. They also associated their feelings of success with understanding the concepts presented in class (88%), which is associated with the Knowledge items from the ML Education Framework, and work on assignments (47%), which is associated with the Skills from the ML Education Framework. One student said that "Personal learning ability progression and ability to utilize the learning from topics to learn even more" was also a factor, which is associated with S6. Independent Out-of-Class Learning (see Figure 4-11b). This focus on project work in boosting students' feelings of success in ML is very much in-line with theories of computational action and constructionism.

Perhaps most importantly with respect to the development of A3. Self-Efficacy, 65% of respondents strongly agreed and 35% agreed with the statement "I will be able to complete an ML project (of a similar level and scale to my final project) on my own." (see Figure 4-12).

Responses to the question "How do you see yourself using the tools, techniques, and methods presented in the class?" reinforced the high A3. Self-Efficacy signals.



(a) Responses to Likert scale question: "How much do you agree with this statement: I felt that I was successful in this class." with standard error bars.



(b) Responses to multiple choice multiple response question: "What did you use to determine your sense of success in this class?" with standard error bars.

Figure 4-11: Likert scale responses to questions relating to feelings of success.

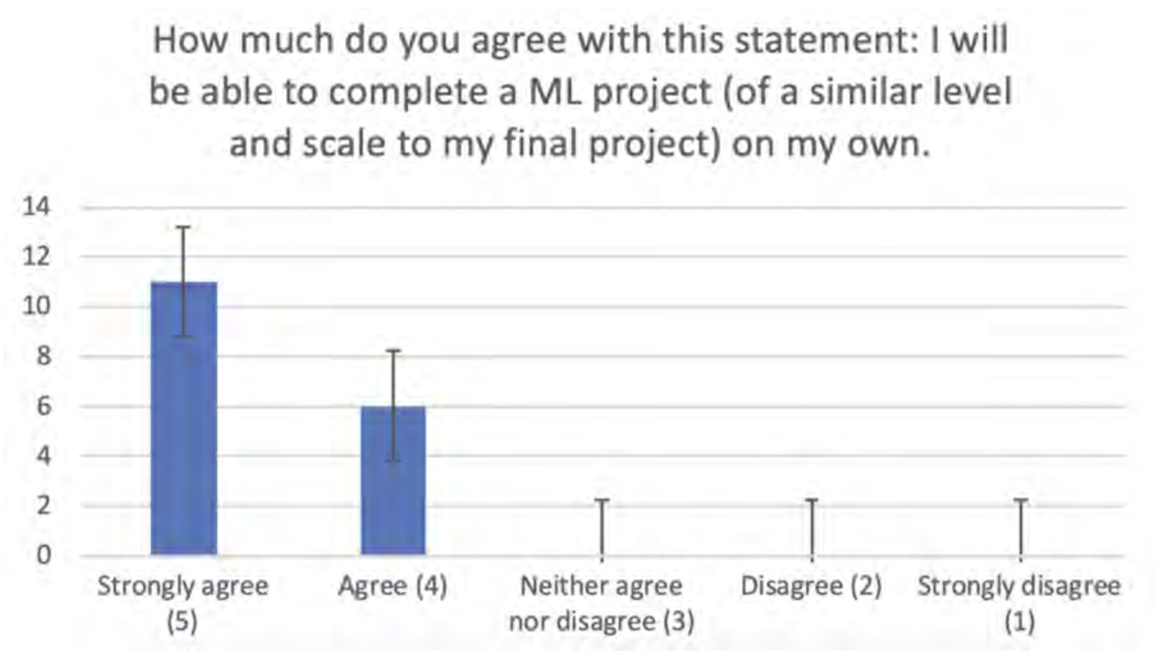


Figure 4-12: Responses to Likert scale question: "How much do you agree with this statement: I will be able to complete a ML project (of a similar level and scale to my final project) on my own." with standard error bars.

The results also indicate that students developed Attitudinal items as well as a sense of usefulness for the Skills they learned through the class. All respondents selected more than one checkbox and none of the respondents chose "I do not care much about using this technology":

- 82% Applying ML to new domains (S3. Creating ML Artifacts)
- 77% Be(ing) able to talk about it with experts (S5. ML Advocacy)
- 77% Being able to talk about it with non-experts (S5. ML Advocacy)
- 65% Using it for fun (A1. Interest)
- 65% Developing my final project further (A4. Persistence)
- 65% Using it as part of an assignment for another class (S3. Creating ML Artifacts, A4. Persistence)
- 65% Using it as part of a job (S3. Creating ML Artifacts, A4. Persistence)

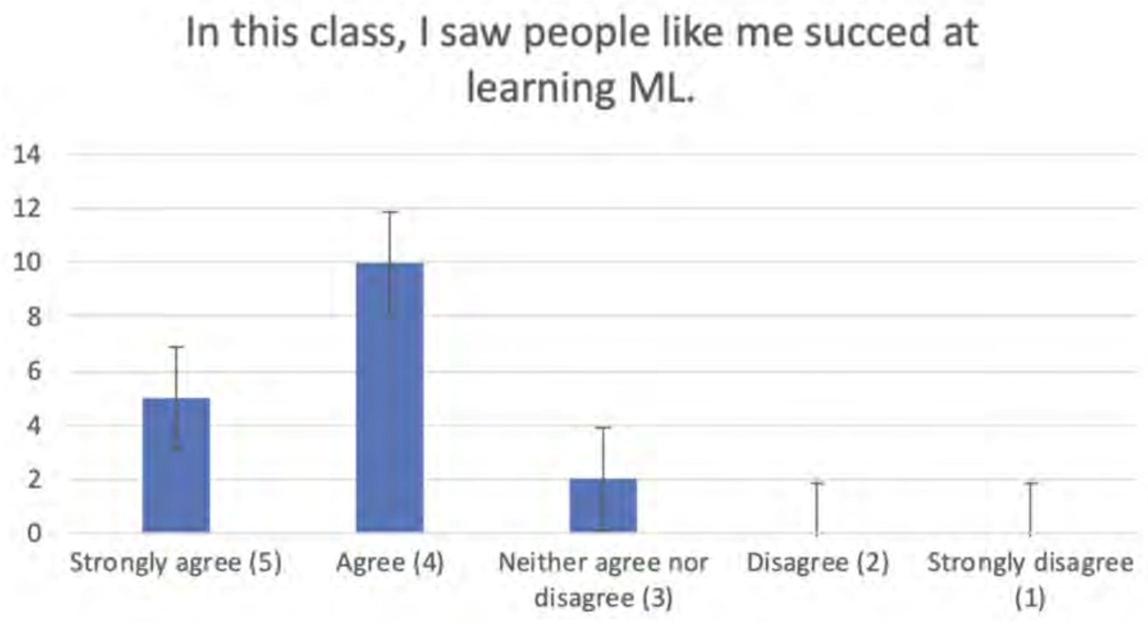
- 35% Developing new ML algorithms and/or architectures (S3. Creating ML Artifacts)
- 0% I do not care much about using this technology

Vicarious Experiences

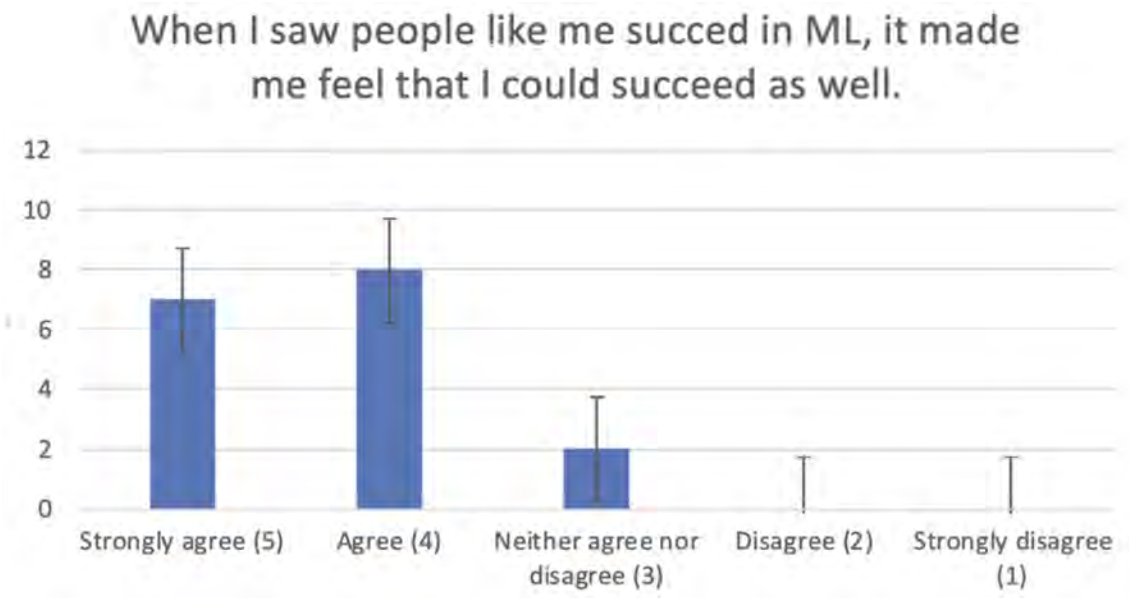
Opportunities for vicarious experiences (or seeing others like oneself being successful) were strategically designed into the course. In-class discussions, project demonstrations, project mentors, and guest speakers served as potential "like me" candidates. Two items in the end-of-course survey captured students' responses to these interventions. When given the item: "In this class, I saw people like me succeed at learning ML," 29% strongly agreed; 59% agreed; and 12% neither agree nor disagree. When given the item: "When I saw people like me succeed at ML, it made me feel that I could succeed as well," 41% strongly agreed, 47% agreed, and 12% neither agree nor disagree. Interestingly, two students who agreed (4) with the first item ("In this class, I saw people like me succeed at learning ML") strongly agreed (5) with the second item ("When I saw people like me succeed at ML, it made me feel that I could succeed as well"). These results indicate that even light vicarious experiences contribute to students' sense of possibility towards ML (see Figure 4-13).

Persuasion and Supportive Messages

Another component of self-efficacy is persuasion, or the extent students feel support and encouragement from "trusted others." Three questions on the end-of-course survey addressed this construct. The first asked students their level of agreement with the statement: "I know I was successful in my project work because other people told me so." 12% of respondents strongly agreed, 53% agreed, 29% neither agree nor disagree, and 1 respondent (6%) strongly disagreed with the statement. Another question asked students their level of agreement with the statement: "Other students in the class helped me learn the technical material." 24% strongly agreed, 47% agreed, and 29% neither agree nor disagree. When asked "If you had a project partner at the



(a) Likert scale responses to: "In this class, I saw people like me succeed at learning ML." with standard error bars.



(b) Likert scale responses to: "When I saw people like me succeed in ML, it made me feel that I could succeed as well." with standard error bars.

Figure 4-13: Likert scale responses to questions relating to in-class peers.

end of the semester, how would you describe their impact on your learning?", 52.9% of students selected "My partner greatly helped with my learning", 23.5% selected "My partner was reasonably helpful to my learning", and 11.8% selected "My partner neither helped nor hindered my learning". No respondent selected "My partner was detrimental to my learning".

Physiological Response

Negative physiological responses to presenting one's work (such as feeling ill or having butterflies, sense of unease) has been shown to be detrimental to one's self-efficacy. The course created a casual, friendly space for exploring new concepts, tools, and ideas, thus minimizing the potential for negative physiological responses. The end-of-course survey asked: "How confident do you feel about describing your project to a non-technical person?" 59% of respondents reported feeling completely confident and 41% reported feeling confident. The second question asked the respondents how much they agreed with the statement: "The project work made me feel uncomfortable." Responses ranged from strongly disagree (59%), disagree (29%), neither agree nor disagree (6%), to agree (6%). The final question in this grouping asked how much students agreed with the statement: "Presenting my project in front of an audience made me feel uncomfortable." 53% of respondents strongly disagreed, 41% disagreed, and 6% neither agreed nor disagreed with this statement.

Students also reported feeling pride in their work: 47% strongly agreeing, 29% agreeing, and 24% neither agreeing nor disagreeing with the statement "I am proud of what I was able to accomplish in my final project." (see Figure 4-14).

4.6.3 Mastery of ML Methods

Eight knowledge concepts (K2. Knowledge of ML Methods) were taught during the modules portion of the course over 7 units. Analysis of lab grades yielded information on the degree of success students had learning the 8 ML topics taught during the modules portion of the course. The instructors deemed that a grade of .6 to .7

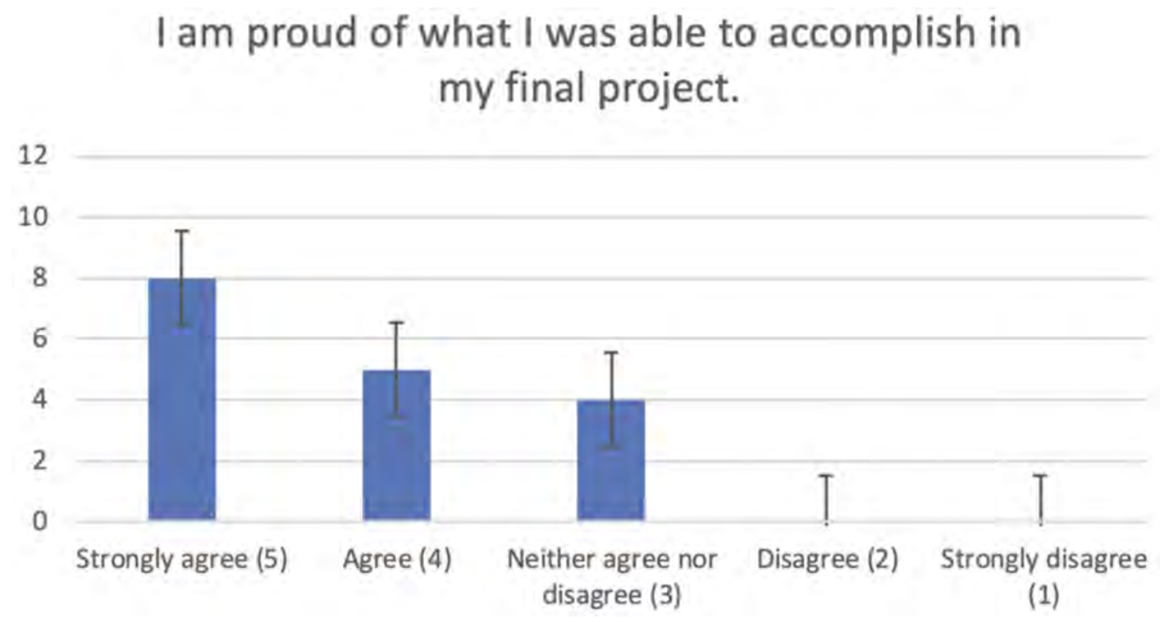


Figure 4-14: Likert scale responses to: "I am proud of what I was able to accomplish in my final project." with standard error bars.

indicated low mastery; .7 to .8 indicated middle mastery; a .8 to .9 indicated high mastery; and .9 to 1.0 indicated expert mastery (see Table 4.4). High standard deviations can be attributed to the number of students who did not complete a substantial portion of the weekly assignment.

The subsequent use of these concepts in capstone projects were indicative of secondary mastery experiences. A top-down thematic approach was used to annotate

Week	Concepts	μ	σ
1	**KNN and Transfer Learning**	0.85	0.28
2	**Multilayer Networks**	0.89	0.17
3	**CNNs** Sub-concept: Multilayer Networks	0.71	0.37
4	**Generative Models and Embeddings**	0.79	0.34
5	**GANs** Sub-concepts: CNNs, Multilayer Networks	0.78	0.32
6	**Reinforcement Learning**	0.80	0.40
7	**RNNs and LSTMs**	0.77	0.37

Table 4.4: Units, core topics addressed, and means and standard deviations of student scores.

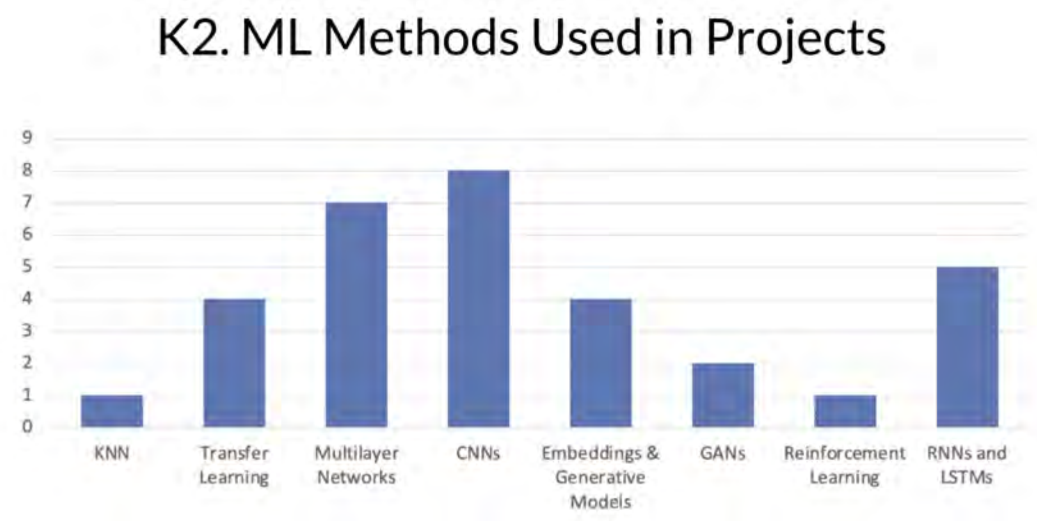


Figure 4-15: Concept used per number of projects.

project write-ups, demos, and presentations for the concepts utilized. All 8 knowledge concepts were represented in at least one project (see Figure 4-15). It is important to note that the project topic strongly influenced which concepts students applied. From this analysis, we present the top 5 core concepts to educating students in actionable ML across various applications: Transfer Learning, Multilayer Networks, Convolutional Neural Networks, Embeddings and Generative Models, and Recurrent Neural Networks and Long Short-Term Memory.

4.6.4 Mastery of ML Artifact Creation Skills

Two post- survey questions assessed students' critical, practical, and action-oriented abilities in ML. When asked how they determine whether or not a problem can and should be solved using ML (S1. ML Problem Scoping), students said that ML could be used when past data is a predictor for future results. Many students also mentioned the importance of data in their response: 11 (65%) described the need for sufficient data and 2 (12%) wrote about data quality, citing concepts such as proper labeling, consistency, and parsability. Five students (29%) wrote that they would look at existing models/research for applicability to a problem. One student each wrote about the ability of the problem to accept error, the need for computational resources, and the importance of ethical considerations. Interestingly, 4 students (24%) wrote that they **would not use ML when the problem could be solved using algorithmic or analytical methods**. Students may have become wary of ML as an inaccurate, unexplainable technique rather than a silver bullet.

Students were asked to describe the general process of creating an ML solution to a suitable problem (S2. ML Project Planning). There were two main methods of attack. The first mode was input/output-centric. Eight students (47%) described a process that focused on defining the data and output before figuring out what architectures and computing resources to use. An example response of this type was: "*acquire as much data as possible -> understand the data (and actually spend a lot of time here) -> clean the data to suit purpose and remove anything negative or skewing and figure something out for missing data -> figure out the model that works -> implement the model using the right architecture. don't forget to use the right loss and a validation data to confirm results -> train the model -> iterate back to architecture a couple times -> publish model somewhere. maybe other ppl can give insights. if one can sit with a professional for any step on this, that'd be great becaue they won't fall for same dumb mistakes*". The second mode was model-centric. Six students (35%) wrote that they first looked at industry standard models or similar solutions from research before experimenting upon the prior work. An example response of this type was: "*Setup the problem such that you know what problem you are trying to solve. Start with the industry standard regarding models, if this problem has been tackled earlier (using CNNs for example) or something similar has been done. Experiment with different architectures, and finally settle on the one that does the best. Take that architecture and then work towards improving the performance*". The rest of the responses mentioned that designing the model can be unpredictable and takes a lot of trial and error.

It was not known in advance which sub-skills taught during the course would be the most useful for capstone project work, so all of the projects were analyzed to identify salient sub-skills. A bottom-up emergent approach was used to annotate sub-skills as evident from project write-ups, demos, and presentations. 21 sub-skills were distilled from the analysis and can be sorted under 6 categories.

The first category is "problem set-up" (correlated to S1. ML Problem Scoping) and includes the skills "scoping a problem" and "modifying a problem": for example, deciding to reimplement and expand on the goals of a research paper.

The second category is "data" and includes the skills "choosing datasets"; "choosing features": isolating specific features of data for training; and "creating datasets": this includes scraping the web to curate a dataset or significantly processing a pre-compiled dataset. These items encompass both S1. ML Problem Scoping and S2. ML Project Planning.

The third category is "architecture" and includes "choosing models" (correlated to S2. ML Project Planning), "modifying models" (correlated to S3. Creating ML Artifacts), and "creating models" (correlated to S3. Creating ML Artifacts).

The fourth category is "hyperparameters" (correlated to S3. Creating ML Artifacts) and includes "choosing optimizers", "modifying optimizers", "choosing loss functions", "modifying loss functions", "creating loss functions", "modifying learning rates", and "modifying batch sizes".

The fifth category is "analysis" and includes "using filters" and "training and testing". These items encompass both S3. Creating ML Artifacts and portions of S4. Analysis of ML Design Intentions and Results.

The sixth category consists of "specific skills for reinforcement learning" (correlated to S3. Creating ML Artifacts) and includes "creating reward functions", "creating an environment", "creating state spaces", and "creating agents".

The level/action descriptor for a sub-skill was based off of the Use-Modify-Create progression with the addition of "Choose". We found that searching and selecting from a set of tools, techniques, and data entails more significant discernment and judgment than simply knowing how to use a model or dataset that is pre-selected and provided. A summary of the number of projects that used each sub-skill is shown in Figure 4-16.

From this analysis, the top 8 core sub-skills found to be essential in educating students to develop self-efficacy in ML are: "scoping a problem", "choosing datasets", "creating datasets", "choosing models", "modifying models", "creating models", "mod-

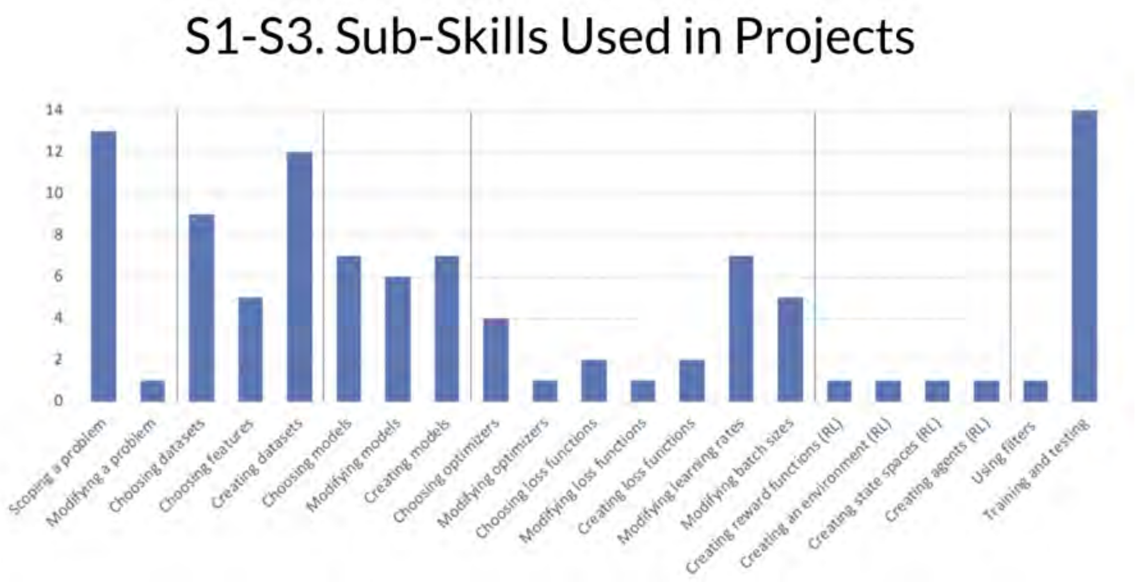


Figure 4-16: Skill used per number of projects.

ifying learning rates", and "training and testing".

4.6.5 Identity and Perceptions

A goal of the course was to help students develop computational identities as members of the ML community. The final writeup for capstone projects was in the form of a blog post, which is commonly used to share knowledge within the ML community informally and quickly. This style of contribution helped students learn the disciplinary norms, situate their work in the larger field, and identify with other ML developers, enthusiasts, and researchers. 13 of the 14 projects from the course shared their project blog posts publicly on the web [70]. Responses from the end-of-course survey showed that students entered the course with limited practical ability as ML developers and left the course feeling that they had the skills to create ML projects. Students also envisioned themselves using the tools, techniques, and methods presented in the class in a variety of future pursuits.

Student responses to "How did your views on machine learning change through taking this course?" generally clustered into 4 categories (see Figure 4-17). 9 students (52.9%) mentioned a **personal realization of the easy application potential of**

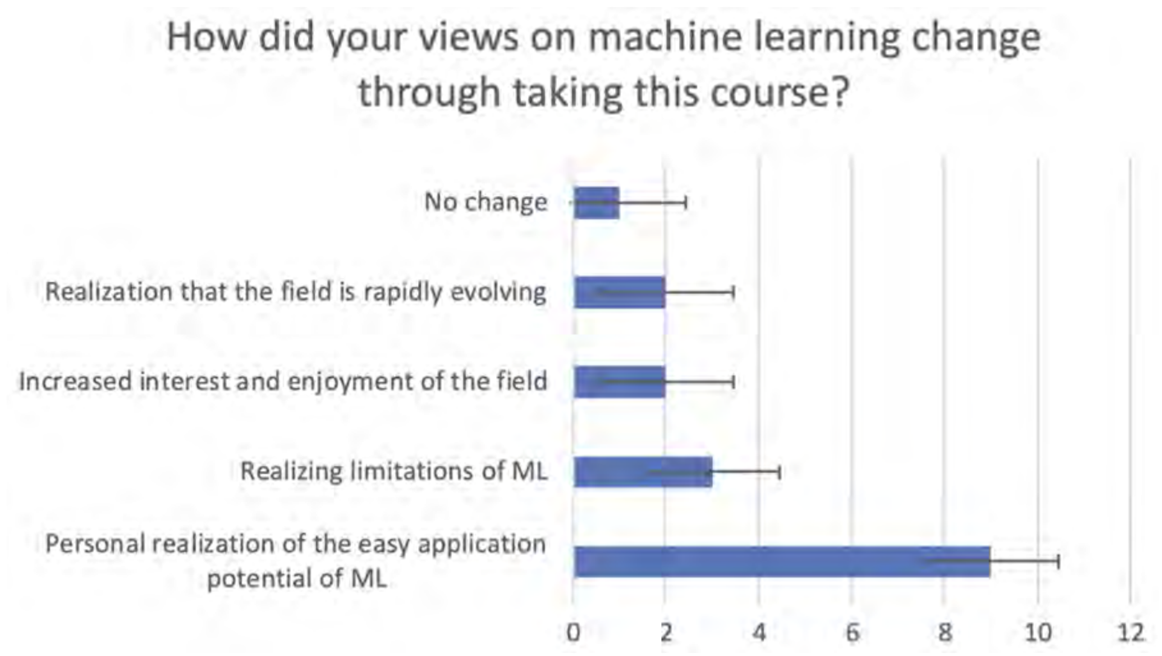


Figure 4-17: Types of student responses to the question "How did your views on machine learning change through taking this course?" with standard error bars.

ML: "*I realized that you can do ML projects even without extensive theoretical or mathematical background... I feel a lot more comfortable in taking on a ML project*". 3 students (17.6%) wrote about **realizing limitations of ML** despite what it can do: "*It's not as easy as plugging in data into an algorithm. one has to understand both their data and the algorithm. I think this is something that's wonderful to learn on my own than it is to learn in the traditional class setting.*" 2 students (11.8%) wrote that the course **increased their interest and enjoyment of the field** and 2 (11.8%) wrote about a realization that **the field is rapidly evolving**. 1 student (5.9%) wrote that their views did not really change.

4.7 Discussion

The Deep Learning Practicum was designed as an introductory ML course that invited college students from a diverse range of backgrounds and preparation to participate in the creation and application of ML. The course aimed to help students from none to novice-levels of CS and ML knowledge develop computational identities as members of the ML community and gain self-efficacy in ML. In general, students highly enjoyed the course, felt that it helped demystify ML, and were encouraged to pursue independent ML projects in the future. Analysis of student feedback and artifacts indicate that successful completion of the capstone projects most heavily influenced student development of self-efficacy and computational identity in relation to ML, and that the modules portion of the course prepared students well for the project assignment as well as future independent projects in ML.

Deep Learning Practicum was taught in TensorFlow.js due to the ease of creating fast-paced, interactive in-class labs, but the core concepts and skills can be taught in any coding language. Students who completed the course expressed both interest and confidence in being able to apply the concepts and skills learned to future ML projects of similar depth and breadth, signifying high A3. Self-Efficacy in ML. The course modules were sufficient to help students gain the skills and knowledge needed to research and apply state-of-the-art ML technologies not covered in class: 4 students (24%) learned new architectures specifically for their project. We hypothesize that the hands-on, exploratory lab work made students feel more comfortable interacting with new architectures and playing with new models. Most importantly, after the course, students felt they could create independent ML applications and were motivated to continue their learning.

We felt that both iterations of the course were successful in our goals, with the second iteration allowing students to learn more from their capstone projects. We believe that the following four components of the course best contributed to its success:

First, while the modules did not teach all of the skills and concepts students needed for every type of ML project, we hypothesize that the demo-focused, exploratory lab work for each topic helped make students feel more comfortable playing with new architectures. This encouraged students to conduct research and learn on their own (S6. Independent Out-of-Class Learning). Every team engaged in S6. to find datasets, models, or techniques for their projects, and 3 teams from the second iteration (21%) applied new ML methods not taught in the class for their final projects.

Second, TensorFlow.js allowed beginners to dive directly into exploring complex and visually appealing ML applications. Being able to modify ML models in the browser allowed for near-instantaneous feedback and reduced infrastructure problems. These fast-paced experiential learning activities helped students quickly gain experience with S3. Creating ML Artifacts.

Third, having designated mentors for each project greatly helped students fast-track their project creation process and helped them start connecting to the larger ML community (A2. Identity and Community). Industry mentors were enthusiastic about working with students and helped them properly scope their projects, find resources, and avoid common mistakes or pitfalls often made by beginners.

Fourth, the informal blog post style of contribution for the final project helped students learn the disciplinary sharing norms, situate their work in the community, and identify with other ML developers, enthusiasts, and researchers. 13 of the 14 projects from the course shared their project blog posts publicly on the web [70].

While these results are promising, we recognize several limitations of this research. The class sizes were small and Massachusetts Institute of Technology students are not representative of the general population. There is also a high staff-to-student ratio due to the laboratory/discussion format of the course as well as the direct mentorship offered for the final projects. This course design can be difficult to replicate due to its high resource commitment. Additionally, tools and technology in machine learning change very quickly, so the applications-focused modules may need to be replaced in the future. While we don't foresee the first few topics becoming irrelevant (weeks 1-3), new developments in research may allow more advanced topics to be replaced. The advanced modules can also be selected based on what type of projects the instructors want to encourage. Finally, we wish to note that we found no significant difference in performance between students of varying levels of ML and coding backgrounds. Many of the students without coding experience also suggested that a version of the course could be adapted for high school students due to the hands-on and un-mathematical nature of the experiential activities.

As a nascent field with a variety of applications, ML should not be considered a tool accessible to only highly trained technical professionals. Policymakers worldwide have recently started considering the implications of ML technology. To ensure reasonable regulation and fairness in the application of ML technology, everyone needs to be engaged in such decisions instead of solely relying on technical experts and legal structures. Thus, we must provide the right educational frameworks to help democratize participation in these pioneering conversations. Computational thinking in the age of machine learning needs to encompass the thinking skills inherent in designing, constructing, training, and testing ML applications.

5

An Analysis of Curricula Using the ML Education Framework

In this chapter, I present a rubric for analyzing ML curricula created from the principles of the ML education framework. The curriculum rubric is intended for use both by curriculum designers who are mapping out the goals of a course they are creating and educators who are choosing what to teach. I apply the curriculum rubric in analyzing the 6 AI and ML courses mentioned in Chapter 1, including Deep Learning Practicum, the course I presented in Chapter 4. I review and compare the results for the 6 courses and offer insights into designing and implementing ML curricula.

5.1 ML Education Framework: Curriculum Rubric

The Curriculum Rubric provides descriptions for each item of the framework on a 0-4 scale, where 0 indicates no coverage of the framework element and 4 indicates high coverage. In addition to the items from the framework, there is one final evaluation item that addresses instructional design, which encourages educators and curriculum developers to think about implementing courses with attention to good pedagogical practices. This rubric has three uses:

- For curriculum designers to highlight which items they want to focus on (used at the beginning of the design process)

- For educators to evaluate how well their curriculum covers elements of the ML education framework (used during the evaluation and redesign process)
- For educators to choose which curricula and workshops they want to use/teach

The guidelines from 4, high coverage, to 1, minimal coverage, of each item of the rubric is presented below. A 0, indicating no coverage, is awarded if the curriculum does not meet the standards for 1. A live version of the Curriculum Rubric for the ML Education Framework can also be found in table form online [69], with the most recent version shown in Appendix A.

- K1. General ML Knowledge
  - 4: Graduates of this course can give a precise definition of machine learning (e.g. "a model that can complete a specific task after being trained by humans to find patterns from large amounts of data") and provide a detailed description of the steps of the machine learning pipeline with technical and socio-ethical considerations for each step.
  - 3: Graduates of this course can give an adequate definition of machine learning and provide a cursory overview of the machine learning pipeline.
  - 2: Graduates of this course can give a superficial definition of machine learning (e.g. "teaching machines to do things that people can do") and provide a few accurate examples of machine learning applications (e.g. Google search, YouTube recommendations).
  - 1: Graduates of this course recognize the term "machine learning" but demonstrate limited abilities in defining the term or providing examples of the technology (e.g. "machine learning has something to do with computers and technology").
- K2. Knowledge of ML Methods
  - 4: Graduates of this course are able to accurately discern when to use a range of machine learning methods across the breadth of the field. They

    are able to describe core technical concepts of these methods and comfortably use/implement them in appropriate applications. (For high school and above, a good range may include supervised methods such as K-nearest neighbors, CART/DT, regression, convolutional neural networks; unsupervised methods such as k-means clustering, principal component analysis, GANs, embeddings; RNNs/LSTMs; reinforcement learning; transfer learning; ensemble methods. For middle school and below, the methods would likely be framed around high level applications that allow students to complete specific tasks using machine learning, including using GAN, RNN, or CNN applications to create art or music, using RL or hidden Markov models to play games...etc.)

  - 3: Graduates of this course are able to accurately discern when to use several machine learning methods. They are able to describe core technical concepts of these methods and use/implement them in appropriate applications.
  - 2: Graduates of this course are able to describe the use cases for and core technical concepts of one or two machine learning methods. They are also able to use/implement the method(s) in appropriate applications.
  - 1: Graduates of this course are able to describe the use cases for and general workings of a few machine learning methods at a surface level.

- K3. Bias in ML Systems
  - 4: Graduates of this course are able to describe how machine learning systems may come to be unpredictably biased against specific groups throughout each step of the machine learning pipeline and critically incorporate the practices of ethical thinking and design in their own work.
  - 3: Graduates of this course are able to describe how machine learning systems may come to be unpredictably biased against specific groups throughout each step of the machine learning pipeline.

  - 2: Graduates of this course are able to detail one or two examples of bias in machine learning systems, describing the causes of bias and resulting harms (e.g. some examples of bias: Northpointe's COMPAS system used in criminal justice, hiring/resume screening models, fruit sorting models performing poorly on green apples vs red ones; some causes: incorrect assumptions, unbalanced training data, proxies, bad benchmarks/metrics, choice of loss function; some harms: hiring discrimination, disparate healthcare outcomes).
  - 1: Graduates of this course are able to articulate that math-based models are not neutral and that it is possible for machine learning systems to create as biased outcomes as human decision-makers or even amplify bias.

- K4. Societal Implications of ML
  - 4: Graduates of this course recognize that it is necessary for creators of ML technologies to consider societal implications of their work and are able to apply ethical and cultural perspectives/concepts to analyses of machine learning artifacts in comprehensive, interrelational, and sensitive ways (i.e. considering privacy, security, potential for abuse, balance of benefits and harms, ethnographic reception, disparate impacts; using tools such as stakeholder analysis, ethical matrix, model cards...etc.).
  - 3: Graduates of this course recognize that it is important for creators of ML technologies to consider societal implications of their work and are able to apply ethical and cultural perspectives/concepts to analyses of machine learning artifacts in not-fully-comprehensive ways.
  - 2: Graduates of this course recognize that it is important for creators of ML technologies to consider societal implications of their work and are able to describe basic/obvious ethical or cultural impacts of ML technology (e.g. ML automating away jobs).
  - 1: Graduates of this course do not recognize that it is important for ML engineers/creators to consider ethical, cultural, or societal consequences of

their work, but are able to articulate that machine learning technologies exist within society and have societal implications.

- S1. ML Problem Scoping
  - 4: Graduates of this course are able to determine which problems can be solved using machine learning from a technical perspective (i.e. whether it is possible to obtain a lot of training data and if the problem is such that past data predicts future outcomes) and which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective (e.g. unsuitable problems: predicting the trajectory of a rocket when it is more easily and accurately done using physics/non-stochastic methods; problems with high standards for explainability or interpretability).
  - 3: Graduates of this course are able to determine which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective but are unable to determine which problems can be solved using machine learning from a technical perspective.
  - 2: Graduates of this course are able to determine which problems can be solved using machine learning from a technical perspective but are unable to determine which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective.
  - 1: Graduates of this course are able to recognize problems similar to exemplar ML problems they have seen in the past.
- S2. ML Project Planning
  - 4: For a range of ML problems, graduates of this course are able to plan out a solution sensitive to both technical implementation considerations (i.e. what kind of data is needed, how to obtain these datasets, what kind of methods/models to use, what computational resources are available for training, what metrics to use for analysis, how the solution fits into the

delivery setting such as mobile vs. cloud, how to maintain a feedback loop into the system after deployment...etc.) and contextual factors including ethics and cultural dimensions (e.g. data privacy, security, how the solution fits into existing human, technical, and systemic processes, potential concerns of users or 'used upon').

  - 3: For a range of ML problems, graduates of this course are able to plan out a solution with an adequate technical approach but minimal contextual considerations.
  - 2: For one or two types of ML problems, graduates of this course are able to plan out a solution with adequate technical approach and contextual considerations.
  - 1: For one or two types of ML problems, graduates of this course are able to plan out a solution with an adequate technical approach but minimal contextual considerations.

- S3. Creating ML Artifacts
  - 4: Graduates of this course are able to accurately identify appropriate machine learning tool(s) that can be used for a wide range of machine learning applications and comfortably use the tool(s) to implement the given application (e.g. Beautiful Soup and Kaggle for data collection; Teachable Machine 2, Cognimates, Keras, and PyTorch for model creation, training, and analysis; AWS and GCP for computation).
  - 3: Graduates of this course are able to identify appropriate machine learning tool(s) that can be used for several types of machine learning applications and use the tool(s) to implement the given application.
  - 2: Graduates of this course are able to identify appropriate machine learning tool(s) for one or two types of machine learning applications and use the tool(s) to implement the given application (e.g. using Teachable Machine to create new classification models or using the Embedding Projector

to analyze different sets of data).

  - 1: Graduates of this course are able to use machine learning tool(s) to replicate one or two machine learning applications from the course (e.g. using Fast Style Transfer webapp to create art).

- S4. Analysis of ML Design Intentions and Results

  - 4: Graduates of this course are able to extract both implicit and explicit design intentions of an ML system (i.e. based on affordances, use cases, accuracy targets of classes/sub-classes, socio-political intentions, financial incentives) and are able to critically analyze those initial intentions in relation to how the deployed system can and should be used (i.e. pros and cons of usage, when you should use something else, societal impacts...).
  - 3: Graduates of this course are able to identify explicit design intentions of an ML system as well as extract a few implicit design intentions, and are able to analyze those initial intentions in relation to how the deployed system can be used.
  - 2: Graduates of this course are able to identify explicit design intentions of an ML system and are able to relate those initial intentions to a few elements of the deployed system.
  - 1: Graduates of this course are able to identify explicit design intentions of an ML system but are unable to relate these intentions to the behavior of the deployed system (e.g. able to identify that the Amazon website wants to sell items to make money, but unable to understand that a product recommender algorithm helps with that intention).

- S5. ML Advocacy

  - 4: Graduates of this course are able to critically engage with their communities regarding machine learning policies, products, and education. They feel motivated to do so and actively seek out opportunities (e.g. teaching family and friends about machine learning, voicing opinions about policies at a town hall).

  - 3: Graduates of this course are able to critically engage with their communities regarding machine learning policies, products, and education. They feel that it is valuable to do so but do not actively seek out opportunities.
  - 2: Graduates of this course are able to critically engage with others regarding machine learning policies, products, or education but do not feel it is important to do so.
  - 1: Graduates of this course are able to engage with others at a surface level regarding machine learning policies, products, or education (e.g. they don't consider social processes or don't recognize ethical problems with ML).

- S6. Independent Out-of-Class Learning
  - 4: During this course, students actively seek out highly engaging learning experiences outside of the classroom (e.g. follow journals or blogs about advances in machine learning or experiment with new products and methods), and are able to independently learn new machine learning concepts and skills.
  - 3: During this course, students actively seek out low-commitment learning experiences outside of the classroom (e.g. keeping up with news about the technology) and are able to independently learn new machine learning concepts or skills.
  - 2: During this course, students do not seek out learning experiences outside of the classroom but are able to learn new machine learning concepts or skills not taught in the course (e.g. if their in-class project requires them to use a method not covered in the curriculum).
  - 1: During this course, students do not seek out learning experiences outside of the classroom but are able to understand and draw connections to

their prior knowledge when presented with information relating to machine learning (e.g. if they see a news article about a machine learning product).

- A1. Interest
  - 4: Graduates of this course are very interested in machine learning and feel highly motivated to engage more with the subject.
  - 3: Graduates of this course are interested in machine learning and feel that they want to engage more with the subject.
  - 2: Graduates of this course have some interest in machine learning and are unsure if they want to engage more with the subject.
  - 1: Graduates of this course have a cursory interest in machine learning and do not plan to engage more with the subject.
- A2. Identity and Community
  - 4: Graduates of this course participate in a larger machine learning community/(ies) where they feel like they belong, can receive help from, and can give back to. They also feel like they have an active role in fostering and shaping the development of the machine learning community/(ies) that they are part of.
  - 3: Graduates of this course participate in a larger machine learning community/(ies) where they feel like they belong, can receive help from, and can give back to.
  - 2: Graduates of this course are aware of at least one larger machine learning tinkerer/learning community but do not participate (i.e. because they cannot access the community, do not feel like they belong, or do not wish to engage).
  - 1: Graduates of this course participate in a small machine learning community of their peers during the course.
- A3. Self-Efficacy

  - 4: Graduates of this course feel fully empowered (highly motivated and prepared) to design and build new and meaningful machine learning artifacts.
  - 3: Graduates of this course feel motivated to design and build new and meaningful machine learning artifacts, but do not feel entirely confident that they are able to succeed without dedicated resources.
  - 2: Graduates of this course feel motivated to design and build new and meaningful machine learning artifacts, but feel that they would not be able to succeed without dedicated resources.
  - 1: Graduates of this course do not feel motivated to design and build new and meaningful machine learning artifacts, but feel that they could potentially succeed if provided with dedicated resources (such as the ones from the course, including mentorship, technical support, compute, software...etc.).

- A4. Persistence
  - 4: Graduates of this course voluntarily and successfully decide to pursue or change career paths in order to work in a job relating to ML.
  - 3: Graduates of this course voluntarily take more classes or do projects in computing, data science, or ML in the future (e.g. middle school students signing up for classes in high school, high school students deciding to major or minor in college) and succeed in these endeavors.
  - 2: Graduates of this course voluntarily engage with machine learning at a surface level in the future (e.g. keep up with news about the technology, experiment with new products).
  - 1: Graduates of this course engage with machine learning in the future out of necessity (i.e. for school or work) but would not if given the choice.

- Instructional Design

- 4: The course was implemented with deliberate attention to good instructional design practices (i.e. well defined expected prior knowledge and course objectives, builds upon a clear learning progression throughout the course, properly links modules with few gaps, activities are engaging and have strong ties to learning outcomes, evaluations are well-defined and helpful to students, provides opportunities for feedback...etc.).
- 3: The course was implemented with adequate attention to good instructional design practices.
- 2: The course was implemented with some attention to good instructional design practices (e.g. thought was given to content sequencing).
- 1: The course was implemented with minimal awareness of good instructional design practices (e.g. material is unsuitable for the audience, unclear learning objectives, activities don't lead to intended learning outcomes).

The curriculum rubric should be applied as appropriate to grade level/audience level instead of as a fixed standard. For example, the following should be considered: Who is the intended audience and what do they know? What is the age range of the audience? Are they trying to apply ML to a specific field or do they want to learn it more generally? Are they not tech-conscious at all, can they navigate around websites and smartphone apps, or can they write computer programs? For each age group/knowledge background, we face the challenge of needing to design different educational interventions. Additionally, it is not expected for a single course to provide advanced coverage for every item of the framework due to time and resource constraints. Curriculum developers can use this rubric to select which items they wish to cover and to what extent based on their goals, the background of their intended audience, and their time and resource constraints.

In the analysis of the 6 AI/ML courses per the curriculum rubric that I discuss in the following sections, I worked with the educators and curriculum developers to identify how their audience corresponds to the 0-4 ratings on the curriculum rubric "as appropriate to grade level". In the future, it would be valuable to work with a wider pool of educators and clarify specific requirements for larger grade bands or audience levels.

5.2 Analysis of 6 ML and AI Curricula Using the Curriculum Rubric

I applied the ML Curriculum Rubric to a set of 6 publicly available AI and ML courses, composed of 2 technical courses aimed at college level students or tech professionals, 2 courses aimed at general adult audiences, and 2 K-12 courses. The technical courses I analyzed are Deep Learning Practicum, per Chapter 4, and Machine Learning by Stanford, which is a popular course offered on Coursera by Andrew Ng [89]. Machine Learning by Stanford is also fairly representative of most college-level technical ML courses. The general adult education courses I analyzed are AI for Everyone [90], which is also a popular course on Coursera run by deeplearning.ai, Andrew Ng's company, and Elements of AI [92], a Finnish course that has been supported by the Finnish government to be released internationally. The two K-12 courses I analyzed are the Technovation AI Families Challenge [31], which is an international 10-week curriculum for primary school students and their families with an emphasis on ideating useful ML projects for their communities, and AI + Ethics Curriculum for Middle School [103], which came from research done by Blakeley Payne in the MIT Media Lab.

I used publicly available lecture materials, video recordings, online texts, problem sets, course reviews, and/or course forums to conduct evaluations of these 6 courses according to the ML Curriculum Rubric. Afterwards, I reached out to the curriculum designers and implementers for each of these courses to discuss my evaluations. I was able to obtain feedback from 5 out of the 6 courses: Deep Learning Practicum, AI for Everyone, Elements of AI, Technovation AI Families Challenge, and AI + Ethics Curriculum for Middle School. The final scores these courses received were agreed upon by both parties.

One limitation of this analysis was that it was difficult across the board to obtain A4. Persistence measurements due to the resources required to follow up with students across several years. Additionally, many public ML and AI courses were made available in the past 2-3 years (the K-12 and general adult audience categories in particular), so in many cases, not enough time has passed to take such measurements. Much of the A4. Persistence scores in this analysis were based on short-term exit surveys or anecdotal feedback that the instructors received.

In the rest of this section, I will be discussing the assignment of scores for each course, as well as overall patterns found through this analysis.

5.2.1 Technical Course: MIT Deep Learning Practicum

As detailed in Chapter 4, Deep Learning Practicum was a full-semester college course offered to MIT students of all grade levels and majors. The course was application and project focused instead of theory-based, and significantly utilized transfer learning. During the first portion of the course, 8 machine learning methods were taught in weekly modules. During the latter portion of the course, students paired up to create capstone projects of their own choosing with support from course staff and dedicated industry mentors. The course met twice a week for 1.5 hours each session for a 15-week semester, resulting in approximately 45 in-class hours. Table 5.1 presents the evaluation for Deep Learning Practicum per the Curriculum Rubric. The evaluation is split into "Intention" and "Result". The instructors and curriculum designers scored their initial intentions for the course separately from the actual post-course results based on feedback from students and analysis of artifacts created by students for the course.

	Intention	Result	Comments
K1. General ML Knowledge	4	4	

K2. Knowledge of ML Methods	3	3	Aside from K-nearest neighbors, Deep Learning Practicum covered deep learning methods and not other machine learning methods.
K3. Bias in ML Systems	4	2	Instructors had hoped that several stand-alone ethics, fairness, and bias lectures during the course would help students gain awareness of how to thoroughly consider such issues during their own ML application design process. However, they found that while students were able to discuss example cases, this level of coverage was not sufficient for students to integrate it into their design processes.
K4. Societal Implications of ML	3	3	
S1. ML Problem Scoping	4	4	

S2. ML Project Planning	4	3.5	From analysis of capstone projects and post- survey responses, significant numbers of students either did not address or lightly addressed contextual considerations in their project planning process, while several project teams focused heavily on social, ethical, or cultural issues. This was also highly dependent on the subject of the capstone project (i.e. A project that tries to build a sign language interpreter may need to consider contextual issues more than a project that tries to train an ML player for a game).
S3. Creating ML Artifacts	4	3	Students developed high levels of expertise in creating ML artifacts relating to the methods needed for their capstone projects. However, they had only adequate mastery over implementing other types of applications taught in the class through the week-long modules.
S4. Analysis of ML Design Intentions and Results	2	2	
S5. ML Advocacy	3	2	While students often discussed ML during the course with their peers and mentors, they did not express having or wanting to have any such discussions with their communities outside of the class.

S6. Independent Out-of-Class Learning	4	4	Most project groups researched and learned how to implement methods or algorithms not taught during the course.
A1. Interest	4	3	From the post- survey, every student could see themselves engaging with ML more in the future. However, several students expressed that their main motivations were for potential career needs and not personal interest in the subject.
A2. Identity and Community	4	2	Instructors had hoped that connecting students to industry mentors and assigning a public blog post for the final project would help students participate in the larger online ML creator/research communities. However, while students heavily participated in the community of their classmates and instructors during the course and were made aware of the online ML community, most only browsed materials that others had posted and did not choose to engage themselves.
A3. Self-Efficacy	4	3	In the post- survey, students agreed that they could complete a ML project of a similar level and scale to their final project on their own, but several expressed that they did not feel that they could succeed in implementing significantly different ML applications without mentorship due to a lack of programming ability.

A4. Persistence	3	3	While we do not have long-term data, every student who filled out the post- survey indicated that they hope to engage in future ML projects or courses.
Instructional Design	4	4	

Table 5.1: Curriculum Rubric evaluation of Deep Learning Practicum, separated by initial "Intentions" and post-course "Results".

5.2.2 Technical Course: Machine Learning by Stanford

Machine Learning by Stanford is offered on Coursera and aims to help students learn machine learning techniques and "gain practice implementing them and getting them to work" [89]. It covers "not only the theoretical underpinnings of learning, but also… the practical know-how needed to quickly and powerfully apply these techniques to new problems". The topics covered draw from case studies and applications, and include "(i) Supervised learning (parametric/non-parametric algorithms, support vector machines, kernels, neural networks); (ii) Unsupervised learning (clustering, dimensionality reduction, recommender systems, deep learning); (iii) Best practices in machine learning (bias/variance theory; innovation process in machine learning and AI)". The course is stated to take approximately 54 hours to complete. Table 5.2 presents the evaluation for Machine Learning by Stanford per the Curriculum Rubric. The evaluation was based on the lectures, problems, and reviews from the offering of the course on Coursera.

	Evaluation	Comments
K1. General ML Knowledge	3	The course did not focus on ethical considerations and bias/fairness when applying ML.

K2. Knowledge of ML Methods	4	
K3. Bias in ML Systems	0	Bias is mentioned in the course in relation to the bias vs. variance context, and not in an ethics or fairness context.
K4. Societal Implications of ML	1	The course discusses many use cases of ML in real-world applications, but does not discuss ethical, cultural, or societal consequences of ML or specific issues that developers should consider.
S1. ML Problem Scoping	2	The course discusses technical impediments to applying ML to a problem as well as many ML methods and use cases, but does not discuss the "should" aspect of this element.
S2. ML Project Planning	3	The course dives deeply into the technical approaches towards building an ML project, but does not discuss contextual considerations.
S3. Creating ML Artifacts	3	While students create many ML artifacts through text-based coding languages in the course, many top reviews on Coursera say that they would not feel comfortable using the tools in the course to implement practical applications. Since this course is a MOOC, these reviews may not be accurate and may not reflect an in-person version of the course.

S4. Analysis of ML Design Intentions and Results	2	The course does not focus on analyzing product intentions but delves deeply into technical analysis and troubleshooting, which should be applicable to this item.
S5. ML Advocacy	1	
S6. Independent Out-of-Class Learning	2	This evaluation is based on how the course encourages students to investigate other methods and learn more about ML outside of the materials provided, but the score is difficult to determine without instructor input or student data and may be inaccurate.
A1. Interest	3	Again, this item is difficult to determine without instructor input or student data and may be inaccurate. Reviews on Coursera were varied. Many reviewers also mentioned that they decided to take the course for perceived career or school needs, which is difficult to disentangle from interest in the subject.
A2. Identity and Community	1	A discussion forum for students of the course was available on Coursera, but the course does not focus on engaging students in a wider ML community of creators.

A3. Self-Efficacy	2	Again, this item is difficult to determine without instructor input or student data and may be inaccurate. Reviews on Coursera mentioned that most students took the course for a perceived career or academic need which points to some level of motivation. However, there were mixed reviews on understanding ML concepts vs. being able to do or make something with ML.
A4. Persistence	Could not evaluate	It was not possible to evaluate persistence based on publicly available information.
Instructional Design	3.5	The course was well designed for its original intended audience of Stanford students, however, many reviewers said that the technical prerequisites of the Coursera offering were unclear, which seems to have made it difficult for some less technical students.

Table 5.2: Curriculum Rubric evaluation of Machine Learning by Stanford.

5.2.3 General Adult Course: AI for Everyone

AI for Everyone is offered on Coursera by deeplearning.ai under "Business Strategy" and is aimed at non-technical audiences who want to learn the business aspects of AI [90]. The course is composed of lectures on 4 topics: "What is AI?", "Building AI Projects", "Building AI in Your Company", and "AI and Society". It covers "The meaning behind common AI terminology, including neural networks, machine learning, deep learning, and data science; What AI realistically can and cannot do; How to spot opportunities to apply AI to problems in your own organization; What it feels like to build machine learning and data science projects; How to work with an AI team and build an AI strategy in your company; and How to navigate ethical and societal discussions surrounding AI". The course is stated to take approximately 6 hours to complete via online lectures. Table 5.3 presents the evaluation for AI for Everyone per the Curriculum Rubric. The evaluation is split into "Intention" and "Result". The curriculum designers scored their initial intentions for the course separately from the post-course results based on feedback but felt that the "Intentions" and "Results" matched fairly well.

	Intention	Result	Comments
K1. General ML Knowledge	3	3	The course thoroughly described machine learning but did not try to dig deeply into the ML pipeline since it was not targeted towards developers.
K2. Knowledge of ML Methods	1	1	The course presented several example use cases and applications of ML commonly used in industry at a surface level.
K3. Bias in ML Systems	2	2	The course gave a few examples of bias and described the effect of biased or non-inclusive datasets.
K4. Societal Implications of ML	2	2	The course lightly covers the potential impact of ML on international economies, governance, and job creation/displacement as well as some ways to resolve potential issues with transition.
S1. ML Problem Scoping	2	2	The curriculum covers the data and prediction aspects of what problems "can" be solved using ML and lightly discusses the "should" aspect in the last unit.

S2. ML Project Planning	0	0	The course is targeted towards business strategists and did not intend to cover the more technical aspects of ML.
S3. Creating ML Artifacts	0	0	
S4. Analysis of ML Design Intentions and Results	2	2	The course presents some analysis of ML products and solutions from a business deployment aspect.
S5. ML Advocacy	1	1	The course touched the topic but the curriculum developers tried to avoid issues of politics to increase the adaptability of the curriculum for wider audiences.
S6. Independent Out-of-Class Learning	3	3	The curriculum developers saw that students actively sought out more information about ML methods and products during the course.
A1. Interest	4	4	
A2. Identity and Community	1	1	Students could participate in a forum of their classmates during the Coursera offering of the course.
A3. Self-Efficacy	2	2	The course is targeted towards non-technical professionals working with/around AI and has a focus on how to build useful tools for business, but doesn't try to build technical confidence through hands-on experiences.

A4. Persistence	4	No data	The curriculum developers hoped that this course would help students implement AI projects in their workplace or encourage them to take further courses in AI but did not have available data on persistence.
Instructional Design	4	4	

Table 5.3: Curriculum Rubric evaluation of AI for Everyone, separated by initial "Intentions" of the course creators and observed "Results".

5.2.4 General Adult Course: Elements of AI

Elements of AI is a free online massive open online course (MOOC) created by the University of Helsinki and Reaktor aimed at "as broad a group of people as possible" who have questions such as "How AI might affect your job or your life?" and who want to "understand how AI will develop and affect us in the coming years" [92]. It has no programming prerequisites, but some basic math knowledge (i.e. fractions, decimals, and arithmetic) is expected. The course "[combines] theory with practical exercises" and is organized into 6 parts: "What is AI?", "AI problem solving", "Real world AI", "Machine learning", "Neural networks", and "Implications". These units are taught through online texts, paper-and-pencil exercises, online interfaces of toy models, and comprehension questions. Each of the 6 parts is expected to take 5-10 hours for an estimate of 45 hours to complete the course. Table 5.4 presents the evaluation for Elements of AI per the Curriculum Rubric. The evaluation is split into "Intention" and "Result". The curriculum designer scored their initial intentions for the course separately from observed results from students and external feedback.

	Intention	Result	Comments

K1. General ML Knowledge	3	3	The examples presented and discussions of problem framing and societal impact give a good overview of ML and the ML pipeline, and the course did not intend to dive too deeply into technical implementation.
K2. Knowledge of ML Methods	1	1	The course provided a few examples of ML methods through specific use cases, but the course designers did not intend to detail specific methods outside of the given use cases or teach students how to implement ML applications.
K3. Bias in ML Systems	2	2	The course gave a few examples of bias in ML and mainly discussed data as a cause of bias.
K4. Societal Implications of ML	3	3	The course provided extensive coverage of societal, cultural, and ethical issues relating to ML (i.e. data permissions, privacy, job displacement). It did not present specific tools or frameworks that students could use in their own practice.
S1. ML Problem Scoping	4	4	
S2. ML Project Planning	0	0	The course did not intend to cover how to implement ML applications from a technical perspective.
S3. Creating ML Artifacts	0	0	

S4. Analysis of ML Design Intentions and Results	3	3	The course provided several example analyses of existing types of ML products from a privacy, bias, and "hype" lens.
S5. ML Advocacy	4	4	The course builds a strong call to action throughout the curriculum to encourage students to engage members of their community in learning about ML. The course developers saw that many former students became course ambassadors and recommended the course to their peer groups. Comments on the Spectrum forum also indicate community outreach. The course developers are planning a user study/impact study to validate these results.
S6. Independent Out-of-Class Learning	3	3	The course includes an activity where students find and write about a related news article. Real-world applications of ML from the units also encouraged students to explore materials outside of the course, which was reflected in student engagement in the Spectrum forum.
A1. Interest	4	4	Student feedback indicated high interest in ML topics from taking the course.
A2. Identity and Community	1	1	The course provided a Spectrum forum for its students but did not connect students to other, larger online ML learning/creator communities.

A3. Self-Efficacy	2	2	The course did not intend to teach students how to build ML applications, although there was student interest and motivation for doing so. The course designers are creating a follow-up course called "Building AI" to teach this topic.
A4. Persistence	3	2.5	There was high variance in persistence due to the nature of the MOOC, but course designers saw that most students either lightly followed ML news or took more courses in ML after the course.
Instructional Design	4	4	

Table 5.4: Curriculum Rubric evaluation of Elements of AI, separated by initial “Intentions” of the course creators and observed “Results”.

5.2.5 K-12 Course: Technovation AI Families Challenge

The Technovation AI Families Challenge is an educational program aimed at children 16 and under along with their families [31]. I evaluated the 2019-2020 season, which has a discussion-based approach to AI along with hands-on activities where students build AI agents using blocks-based or drag-and-drop tools. Students are challenged to ideate AI solutions to problems in their own communities and are provided a network of mentors that they can access through an online forum. The curriculum has 10 lessons: “1. Become an AI Agent Expert, 2. Create a Pattern-Finder AI Agent, 3. Strengthen an AI Agent with Data, 4. Compete to Train Your ML Agent, 5. Brainstorm Your AI Invention, 6. Plan Your AI Invention, 7. Become an Expert for Your Plan, 8. Build Your AI Invention, 9. Pitch Your AI Invention, and 10. Prepare to Submit [the invention to the Technovation competition]”. The first portion of the curriculum focuses on technical topics such as how AI works, where it is currently used, and how it is created, whereas the latter portion of the curriculum has students go through a rigorous ideation process for a personal AI invention. Families with children between 8-15 are encouraged to submit their invention to a yearly pitch competition. The entire curriculum takes approximately 20 hours to complete. Table 5.5 presents the evaluation for the Technovation AI Families Challenge per the Curriculum Rubric. The evaluation is split into "Intention" and "Result". The instructors and curriculum designers scored their initial intentions for the course separately from the actual post-course results based on feedback from students and analysis of artifacts created by students for the course.

	Intention	Result	Comments
K1. General ML Knowledge	3	2	The Technovation curriculum has a broader AI focus, with Lesson 4 on ML. The 2-hour ML unit involved mostly playing with and modifying pre-existing code and datasets, which the curriculum developers felt did not result in more than superficial knowledge of ML.
K2. Knowledge of ML Methods	2	1	The curriculum developers felt that most students did not engage enough with technical concepts of ML methods to thoroughly understand them, but could explain ML on an input/output level.

K3. Bias in ML Systems	1	0.5	There was a discussion activity in Lesson 6 about the consequences of ML inventions, which lightly touched on bias, but based on the results, the curriculum developers felt that the connection to bias could be clarified (they will modify the curriculum for the following season to include a concrete ethics component).
K4. Societal Implications of ML	3	3	
S1. ML Problem Scoping	4	3	Emphasis on community discussion during the project ideation phase of the curriculum was strong on the "should ML be used" component of scoping. Students were weaker on the "can ML be used" component due to shorter interactions with technical ML concepts.
S2. ML Project Planning	2	2	The curriculum developers saw that students become very focused on the end product/prototype due to the competition-based nature of the curriculum. These students may not fully reflect on their learning or design processes until after the program.
S3. Creating ML Artifacts	2	2	Students used a few public ML tools (i.e. Cognimates and MachineLearningforKids) to create ML classification applications.

S4. Analysis of ML Design Intentions and Results	2	2	
S5. ML Advocacy	4	4	The curriculum had a strong built-in community outreach component that involved explaining ML to and discussing students' ML invention ideas with friends, family members, and other community members. In addition to advocacy within these mostly non-expert community groups, students also pitched their inventions to expert mentors through an online forum and discussed their ideas with peers and mentors alike.
S6. Independent Out-of-Class Learning	3	3	The course developers saw that students and their families became more "tuned in" to news about ML and AI technologies during the course.
A1. Interest	4	4	Based on a post- course survey.
A2. Identity and Community	4	4	Students and their families were very involved in the Technovation online forum during the course and continued to give back and engage in that community after completing the program. Additionally, many parents were inspired by the course and created their own ML learning and discussion groups within their local communities.

A3. Self-Efficacy	4	3	The curriculum developers felt that most students did not gain enough technical skills from the program to successfully create technical ML artifacts without mentorship.
A4. Persistence	3	3	No long-term data is available, but responses from the post- course survey indicated that most students planned to engage more with ML in the future.
Instructional Design	4	4	

Table 5.5: Curriculum Rubric evaluation of Technovation AI Families Challenge (2019-2020 season), separated by initial "Intentions" of the course creators and observed "Results".

5.2.6 K-12 Course: MIT AI + Ethics Curriculum for Middle School

"An Ethics of Artificial Intelligence Curriculum for Middle School Students", also known as AI + Ethics Curriculum for Middle School, is a mainly paper-and-pencil/unplugged curriculum with five learning objectives: 1. "Understand the basic mechanics of artificial intelligence systems"; 2. "Understand that all technical systems are socio-technical systems… [which] are not neutral sources of information"; 3. "Recognize there are many stakeholders in a given socio-technical system and that the system can affect these stakeholders differentially"; 4. "Apply both technical understanding of AI and knowledge of stakeholders in order to determine a just goal for a socio-technical system"; and 5. "Consider the impact of technology on the world" [103]. In this course, students design paper prototypes of real-world ML and AI products such as YouTube, discuss the potential effects of ML and AI technologies on different populations, and create ethical matrices. The entire curriculum takes approximately 13 hours to complete. Table 5.6 presents the evaluation for AI + Ethics Curriculum for Middle School per the Curriculum Rubric. The evaluation is split into "Intention" and "Result". The curriculum designer scored her initial intentions for the course separately from the actual post-course results based on feedback from students and analysis of artifacts created by students for the course.

	Intention	Result	Comments
K1. General ML Knowledge	2	2	
K2. Knowledge of ML Methods	1	1	The goal of the course was not to try to teach specific ML methods, but to expose students to a few common methods in order to develop some understanding of ML for analytical activities.
K3. Bias in ML Systems	2	2	Students learned about case studies of bias in ML, focusing on dataset bias, but did not cover the ML pipeline.
K4. Societal Implications of ML	4	4	The course focused on the societal implications of ML technologies for different stakeholders. It provided specific examples in class and gave students political, ethical, and cultural tools/frameworks to use for analysis.
S1. ML Problem Scoping	3	3	The course focused on contextual considerations of technological products but did not dive deeply into technical implementation details.
S2. ML Project Planning	0	0	

S3. Creating ML Artifacts	2	2	The Teachable Machine and GANart activities apply the "I Do - We Do - You Do" Gradual Release of Responsibility Model [33], allowing students to feel an increasing sense of ownership over ML artifacts of their own creation.
S4. Analysis of ML Design Intentions and Results	4	4	During the course, students analyzed several products and applications of ML through multiple iterations of in-class discussions, creating ethical matrices for various stakeholders, and paper-and-pencil redesigns.
S5. ML Advocacy	4	3	The course included strong advocacy components during discussion, socio-technical analysis, and redesign activities, but did not build in community outreach outside of the classroom (i.e. writing letters to representatives) due to schools having liability concerns over political content.
S6. Independent Out-of-Class Learning	3	3	Throughout the course, students contributed daily to a classroom mural where they wrote down two hopes for AI, concerns about AI, questions about AI, or feelings/drawings of AI. These contributions indicated that students furthered their learning at home by talking to their parents, researching politicians' knowledge of AI, and looking into applications of ML that were not covered in the course.

A1. Interest	3	3	Feedback after the course indicated that students became more interested in the technical aspects of ML through first engaging with the societal impact of the technology.
A2. Identity and Community	1	2	Students engaged heavily with the community of their peers during the course. The instructors also included several additional activities not in the curriculum that increased students' sense of connection with a wider ML community: inviting engineers from nearby ML companies to come to talk about their work, having students participate in an open house where they presented their projects to MIT students and staff. These additional activities may be difficult to replicate in other settings but may be possible to replace to a certain extent with videos.

A3. Self-Efficacy	4	3	This course targets self-efficacy in being able to critically analyze technology in general instead of targeting self-efficacy in creating ML applications. At the beginning of the course, the instructors saw that students felt powerless about being able to influence the designs of dominant tech products. However, after the YouTube Redesign activity and seeing the many redesigns of Twitter, students gained more confidence.
A4. Persistence	3	3	From short to medium term feedback, many students worked on more ML projects at home with their parents, taught Teachable Machine to a friend or sibling, and decided to sign up for ML classes.
Instructional Design	4	4	

Table 5.6: Curriculum Rubric evaluation of AI + Ethics Curriculum for Middle School, separated by initial "Intentions" of the course creators and observed "Results".

5.3 Discussion

In this subsection, I will discuss some trends from my evaluation and conversations with other curriculum developers. I will be plotting the evaluation results for each category of ML/AI courses (technical, K-12, and general adult audience) on radar charts. For courses where both "Intention" and "Result" were collected, the "Result" scores are plotted. Only two courses were evaluated per each of the three categories, so these results are not statistically significant. However, the courses selected for evaluation are exemplary for their category and can provide insights toward overall trends in ML education.

5.3.1 K-12 vs. Technical ML Courses

The radar charts for the evaluation results of the K-12 and technical ML courses are presented in Figure 5-1. The chart is on a 0-4 scale per the ML Education Framework Curriculum Rubric with the four "Knowledge" items marked as K1-K4, the six "Skills" marked as S1-S6, and the four "Attitude" items marked as A1-A4. The three knowledge items and three skills that are minimally necessary for courses for ML-engaged citizens (see Section 3.1.4) are marked with an asterisk *.

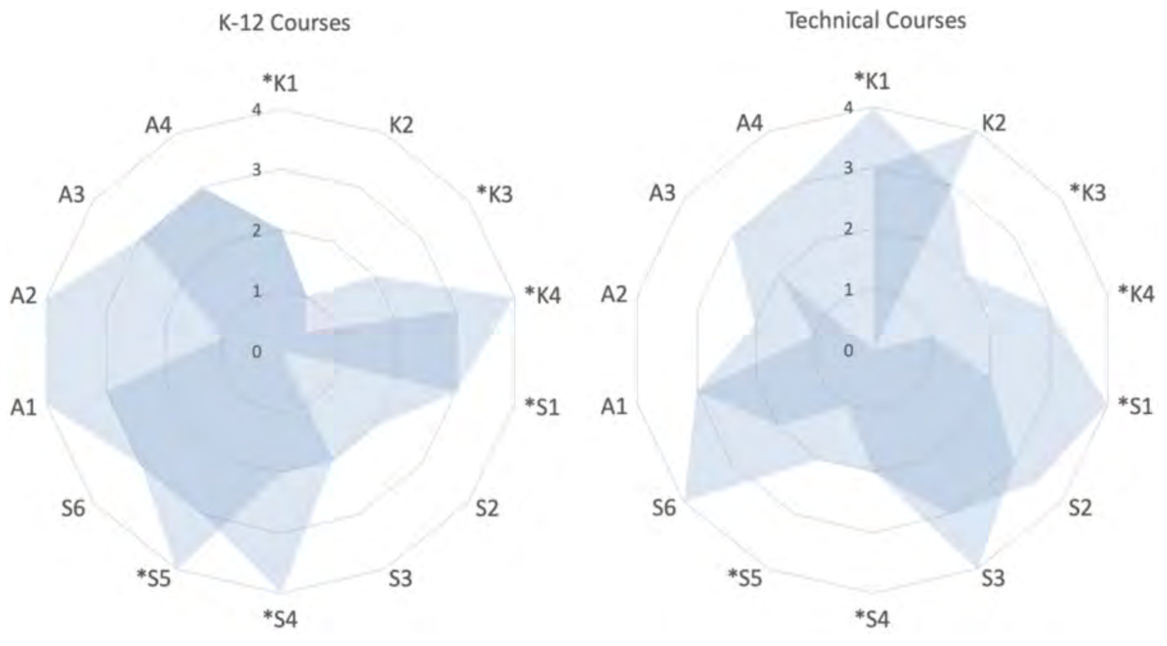


Figure 5-1: Evaluation results for two ML and AI K-12 courses per the ML Education Framework Curriculum Rubric (left) and evaluation results for two ML technical courses (right). The asterisk * elements are the minimally required items from the ML Education Framework for ML-engaged citizens.

The two radar charts for K-12 and technical courses are nearly reversed/opposite in a complementary way. In terms of Knowledge and Skills, the focus of K-12 courses seemed to be on K4. Societal Implications of ML, S4. Analysis of ML Design Intentions and Results, and S5. ML Advocacy. The technical courses had significantly lower coverage of these 3 Knowledge and Skill items and instead delve deeply into items associated with being able to create technical ML artifacts: K1. General ML Knowledge, K2, Knowledge of ML Methods, S1. ML Problem Scoping, S2. ML Project Planning, S3. Creating ML Artifacts, and S6. Independent Out-of-Class Learning. K-12 courses also focused more on all four attitudinal items overall compared to technical courses. In particular, K-12 curricula seem to have more built-in learning experiences for developing A1. Interest and A2. Identity and Community.

These results reflect the different motivations of K-12 vs. technical ML courses. Most K-12 educators aim to engage a wide range of students in ML/AI so that they are not intimidated by it, so that they are motivated to learn more about it in the future, and so that they have the basic knowledge and skills necessary to effectively use and interact with everyday ML products in their lives. The main goal of technical ML educators is to help students who already have some background in programming or advanced mathematics gain the ability to work on ML engineering projects or research in ML. Perhaps due to the nature of students who seek out these technical ML courses, there is already an established interest towards ML (e.g. career motivations) so instructors do not feel the need to spend time on developing that item of the framework. However, it may be useful for technical ML courses to develop A2. Identity and Community more in their curriculum.

From educational theories and research, in order for students to continue independent learning after an intervention or to create more ML artifacts, an ML learning/creator community that students can receive help from and give back to is very valuable. The students of technical courses tend to be older than K-12 audiences, so they have more ability to seek out these communities on their own. However, the ML and AI field lacks diversity, which can make these communities less welcoming for students from marginalized communities [39]. It is important for the entire ML/AI field to have more ML creators from diverse backgrounds in order to identify and solve a wider range of problems [117], and helping students experience positive interactions with these larger ML learning/creator communities during the more controlled environment of a technical ML course may help smoothen this transition.

5.3.2 General Adult ML and AI Courses

The radar chart for the evaluation results of the two ML/AI courses for general adult audiences is presented in Figure 5-2. The chart is on a 0-4 scale per the ML Education Framework Curriculum Rubric with the four "Knowledge" items marked as K1-K4, the six "Skills" marked as S1-S6, and the four "Attitude" items marked as A1-A4. The three knowledge items and three skills that are minimally necessary for courses for ML-engaged citizens (see Section 3.1.4) are marked with an asterisk *.

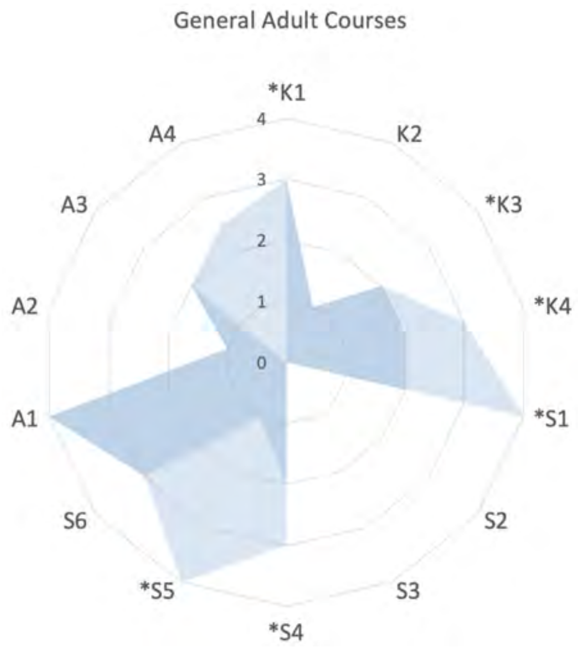


Figure 5-2: Evaluation results for two ML and AI courses for general adult audiences per the ML Education Framework Curriculum Rubric. The asterisk * elements are the minimally required items from the ML Education Framework for ML-engaged citizens.

Compared to K-12 or technical ML courses (see Figure 5-1), the general adult education courses I analyzed covered much less material overall. The focus of this category of ML education seems to be on A1, fostering interest in ML, with some direction for S1, how to scope an ML problem. These courses tried to serve as a brief introduction to ML and pointed students to technical ML courses if they were interested in further independent learning. There were minimal hands-on experiential learning activities with ML tools where students could create ML artifacts. There also seemed to be minimal attempts to develop identity and community.

Based on my conversations with the curriculum creators, general adult AI and ML education seems to have two types of audiences: 1. Adults who are curious or concerned about the field and wish to learn more; and 2. Companies who wish to provide their non-technical workforce some level of ML education to prepare for future work relating to the field. The first audience sub-category is similar to K-12 audiences and it may be possible to modify tools and curricula from that category. The second audience sub-category may be more interested in curricula with narrower focus for specific businesses. In either case, general adult audiences usually have fewer obvious incentives compared to K-12 or technical audiences— they are likely to be learning mostly out of interest instead of to succeed at a class in school or to gain skills that are immediately applicable to their work— so these audiences usually wish to spend less time on a full ML or AI course.

5.3.3 Future Development of the Curriculum Rubric

During the conversations with K-12 curriculum developers, we discussed several ideas for future development of the Curriculum Rubric that would make it easier to use for K-12 educators and course developers. Educators felt that it would be valuable to work with K-12 teachers to clarify more detailed expectations for the 1-4 scores of each item of the framework for each grade band. It may be possible to pivot this rubric to work with standard educational guidelines such as the standards that the AI4K12 Initiative is developing if we can identify resources and tools that pair well with items of the framework at each grade band.

AI4K12 is a group in the United States that is working on a set of nationwide educational standards for AI (see Section 2.1.3). Their work is based upon a set of 5 Big Ideas, one of which is Machine Learning. They list out what students from each grade band should be able to do in relation to each of their 5 Big Ideas. For Big Idea #3: Machine Learning, they have listed 2-3 items for each of their 4 grade bands for K-12 [37]. While my framework can be thought of as a rubric with a 0-4 scale for each item of the framework, the AI4K12 guidelines are organized as specific "do" activities that serve as benchmarks for student learning. The motivations behind the AI4K12 standards and my ML Education Framework are also different. AI4K12 aims to establish essential items that every student should know and be able to do across all of the classrooms in the US, where there are huge disparities in what resources are available. My ML Education Framework is trying to address the more general process of transforming ML consumers to ML-engaged citizens, and then onwards to tinkerers and eventually researchers or engineers.

Despite these differences, many activities from the AI4K12 guidelines correlate nicely onto items in my ML Education Framework. For AI4K12's Big Idea #3: Machine Learning, their 9 "do" items [37] directly relate to 4 elements of my framework:

- Grades K-2:
  - Learn from patterns in data with 'unplugged activities' (**K1: General ML Knowledge**)
  - Use a classifier that recognizes drawings. Use Google Autodraw or Cognimates Train Doodle to investigate how training sets work to identify images and discuss how the program knows what they are drawing (**S3: Creating ML Artifacts**)
- Grades 3-5:
  - Describe and compare the three different machine learning approaches: supervised, unsupervised and reinforcement learning (**K2: Knowledge of ML Methods**)
  - Modify an interactive machine learning project by training its model (**S3: Creating ML Artifacts**)
  - Describe how algorithms and machine learning can exhibit biases (**K3: Bias in ML Systems**)
- Grades 6-8:

    - Identify bias in a training data set and extend the training set to address the bias (**K3: Bias in ML Systems**)
    - Hand-simulate the training of a simple neural network (**K2: Knowledge of ML Methods**)

- Grades 9-12:
    - Train a neural network (1-3 layers) with TensorFlow Playground (**S3: Creating ML Artifacts**)
    - Trace and experiment with a simple ML algorithm (**K2: Knowledge of ML Methods**)

While the AI4K12 is still developing their lists of standards for each Big Idea and grade band, additional items from my framework are likely to be covered by the other 4 Big Ideas from AI4K12, in particular, K4. Societal Implications of ML in Big Idea #5: Societal Impact. It seems useful to work in collaboration with similar organizations in the future, which will probably require working closely with educators to detail specific expectations per grade band and working with tool developers to create public ML tools that pair well with each grade band and application.

6

Public ML Tools and PIC, the Personal Image Classifier

As discussed in Chapter 5, in order to teach AI and ML curricula to more general and less technical audiences, we must also create tools that allow for hands-on exploration of ML that pair well with each grade band and application. There exist many ML "demos" that allow students to play with ML applications in contained settings to explore what ML can do and the limitations of ML (e.g. Sketch-RNN allows users to draw with their mouse and is able to complete the drawing via an ML model [42]; MuseNet can generate musical riffs based on user-specified styles and instruments [105]). These ML *demos* are very suitable for the "Use" portion of the ML learning progression, but *tools* that provide students with opportunities to build ML models and applications for themselves are necessary for students to move towards the "Modify" and "Create" stages of their learning. I refer to these tools as "public ML tools" and they play a crucial role in the ML education community. Public ML tools are essential to the learning experience for students to engage more deeply with ML in a hands-on way or create ML artifacts. What public ML tools are available also affect what curricula can be designed and taught.

The highest utility and highest difficulty tools for machine learning are text-based ML libraries such as PyTorch [101] or TensorFlow [1]. These types of tools are most often used by researchers or engineers, and are taught in technical ML courses.

Many public ML tools for youth and general audiences have sprung up in the past 2-3 years. These include popular graphical interfaces such as Teachable Machine or Model Builder, which we utilized in Deep Learning Practicum, that help people explore machine learning concepts through training models. There are also webapps that allow people to creatively play around with ML such as through style transfer or GAN Paint [9]. Blocks-based programming tools have also been built to teach machine learning within the context of programming concepts (i.e. ecraft2learn [63] and Cognimates).

In this chapter, I present the Personal Image Classifier (PIC), a public ML tool for high school and above students to train and test their own image classification models. Image classification was chosen as the focus of this public ML tool because of its clear inputs and outputs.. The general public is also able to conceptualize potential uses for image identification due to the ubiquity of exposure through real life applications. Similar to Teachable Machine and some other ML exploration tools, students are able to personalize PIC by using images taken from their webcams or downloaded onto their computers and defining the labels. PIC extends these applications, however, through a connection with MIT App Inventor [56], a blocks-based programming language for creating mobile apps. App Inventor is widely used by millions of people across the world to easily build a wide range of highly customizable mobile applications. Its audience includes children as young as middle school to adults across 195 countries. The PIC webtool is hosted at https://classifier.appinventor.mit.edu

The main artifacts built for the PIC project are a web interface that allows users to train custom image classification models, and an MIT App Inventor extension that allows users to upload their image classifiers for use in mobile applications. These tools allow a user to easily create image classification models and use them in mobile applications that they build themselves (see Figure 6-1 for the workflow). Most of the material in this chapter has been previously published [121, 120].

In the PIC webapp, users specify items for the image classifier to learn and provide examples using the webcam or uploaded from their laptop (see Figure 6-2). The web interface allows users to train the classification model on these images and labels, test

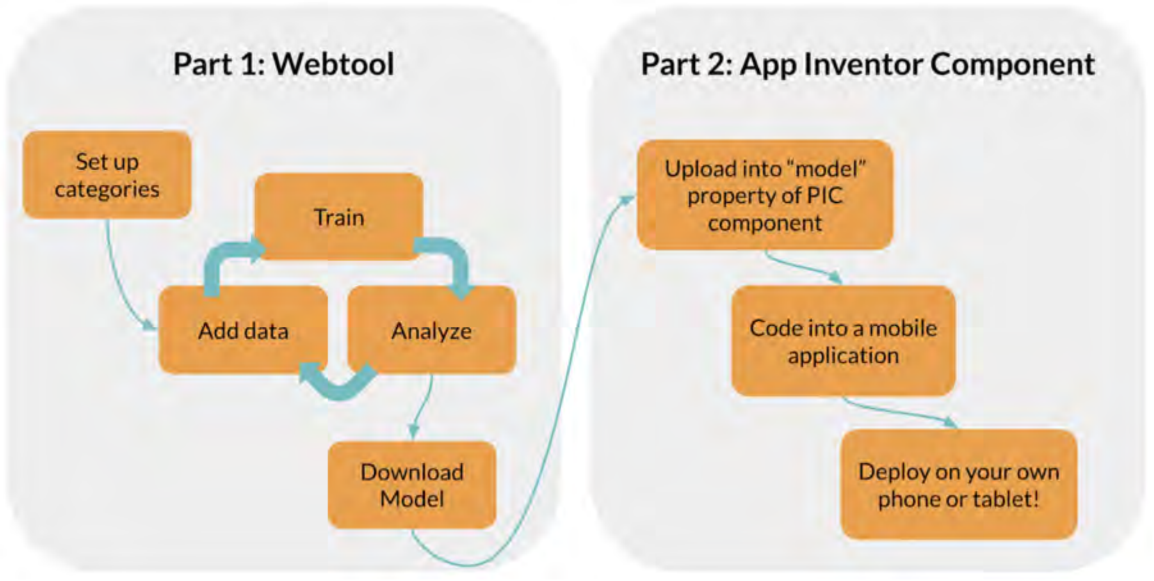

Part 1: Webtool
Set up categories
Train
Add data
Analyze
Download Model
Part 2: App Inventor Component
Upload into "model" property of PIC component
Code into a mobile application
Deploy on your own phone or tablet!

Figure 6-1: The PIC webtool allows users to train a personal image classifier, which they can port into the MIT App Inventor extension for mobile application creation.

and analyze the model's performance, and download the trained model.

In the MIT App Inventor extension, users can upload the downloaded model from the webapp through one of the extension's properties (see Figure 6-3), and then construct the desired mobile application using the extension's provided blocks.

In the rest of this chapter, I will detail:

1. The technical implementation of the PIC webtool and features built into PIC that allow users to analyze the behavior of their models.
2. An extension for MIT App Inventor that enables users to incorporate image classification models trained on the PIC webtool into mobile applications.
3. High school level curricula for workshops to teach basic machine learning principles based on PIC and the MIT App Inventor extension.
4. Results and discussion from usability studies with two classes of high school students from Boston Latin Academy.

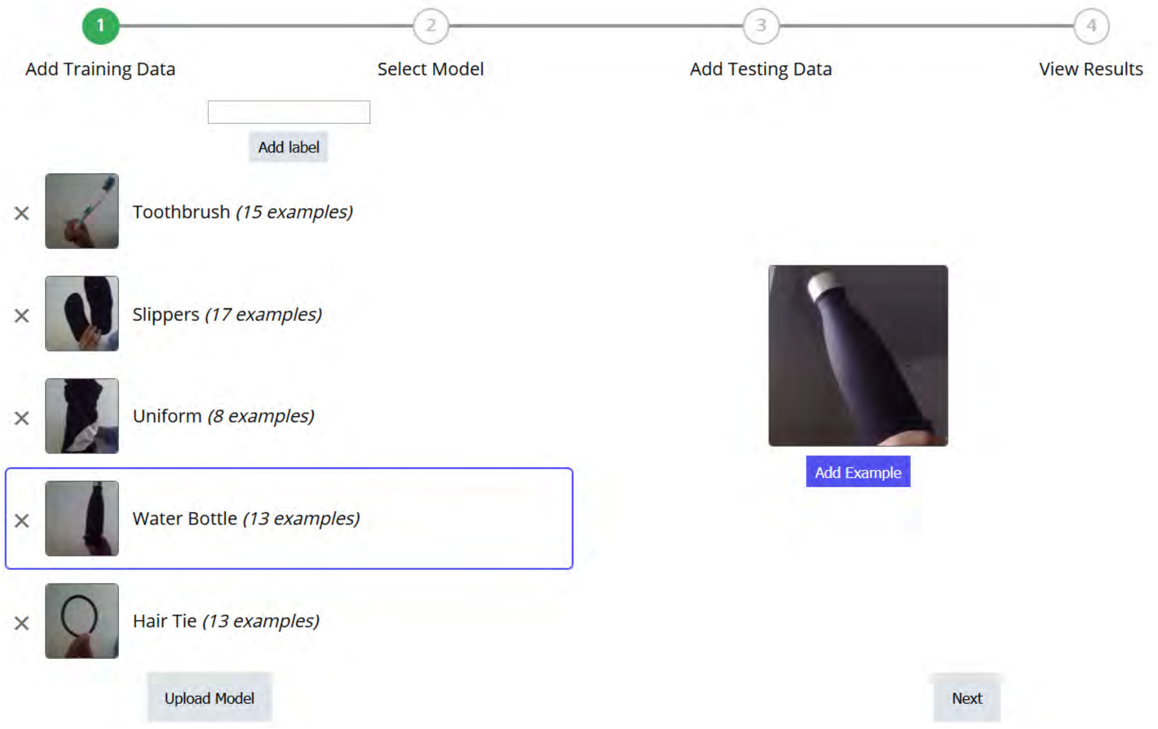


Figure 6-2: The PIC web app allows users to easily create image classification models to recognize objects. The only task required is to specify the objects as labels and take example pictures of the objects using their webcams.

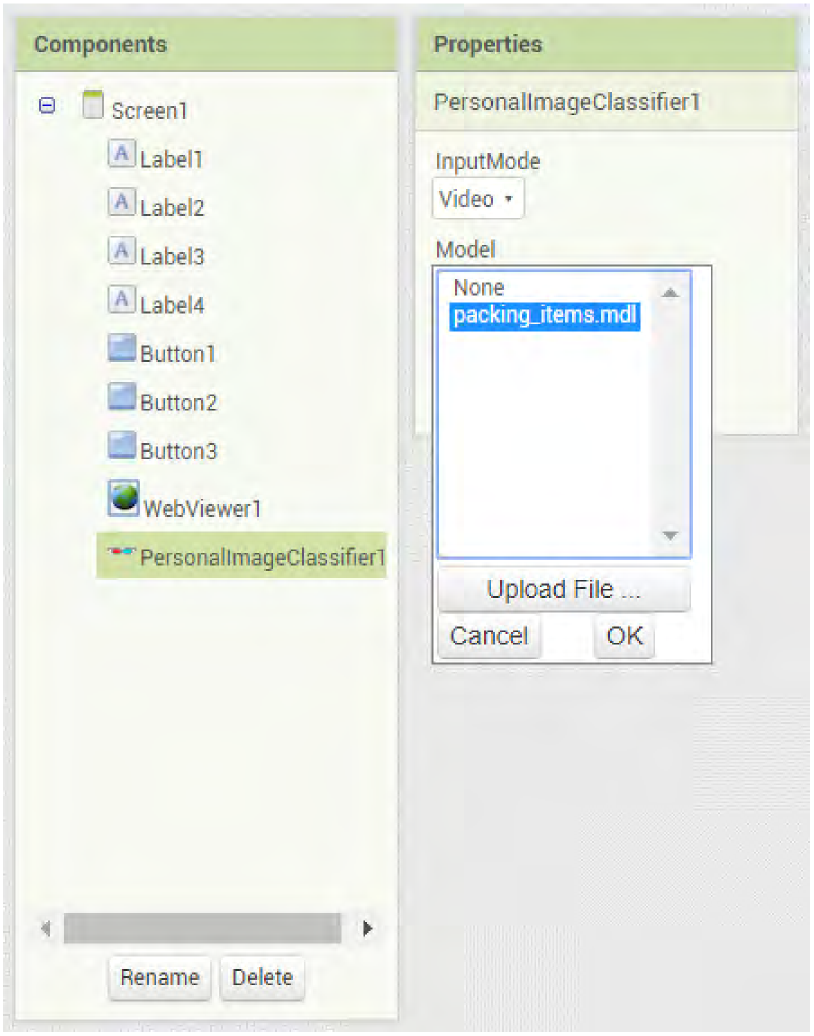


Figure 6-3: The PIC extension for MIT App Inventor enables users to upload their trained models through the extension's properties. The blocks provided by the extension will utilize the uploaded model to perform classifications.

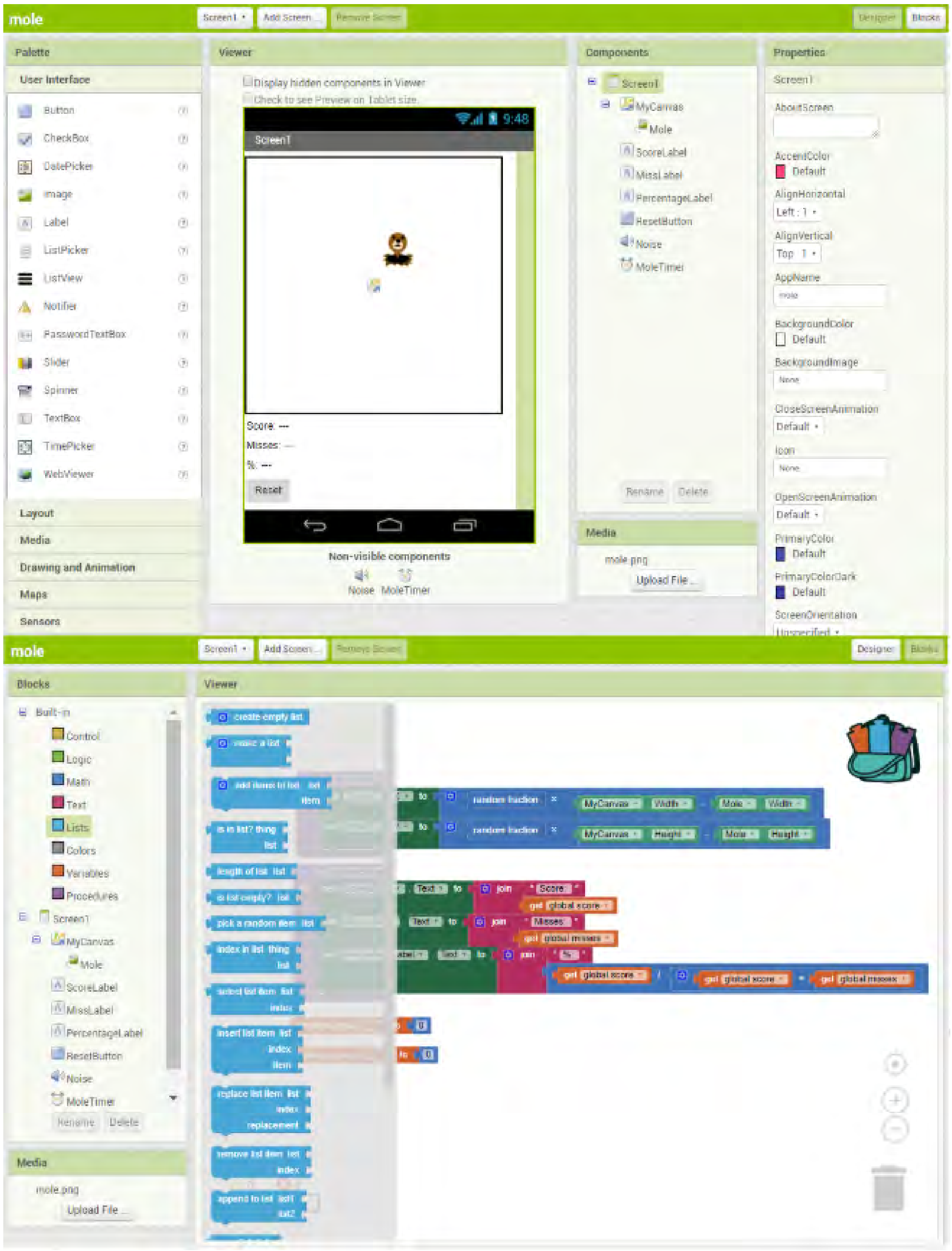
6.1 MIT App Inventor

MIT App Inventor is a free and open-source web platform that allows users to create mobile applications through a blocks-based programming language and high level abstractions (see Figure 6-4). Since its development began in 2009, the primary goal of MIT App Inventor has been to democratize the creation of mobile applications by allowing people with little to no programming experience to create highly functional apps [56]. Through MIT App Inventor, users can build apps that interface with sensors in their smartphone and the web as a whole, including the smartphone camera, accelerometer, text messaging, voice calls, external Bluetooth/IoT devices, and internet connectivity. Due to its low barrier of entry, MIT App Inventor is used as the primary teaching tool for many introductory programming classes and workshops targeting students anywhere from late elementary school to university and professional level. This web interface is available worldwide, with 8.2 million registered users from 195 countries that have created a total of 34 million applications. MIT App Inventor's accessibility to young students and popularity as an international computer science teaching tool makes it a great platform for introducing applicable ML to a wide audience.

In the base web application, MIT App Inventor users can drag in components that interface with the internet and features of their mobile devices. For example, there are components for making API requests and for using the phone's camera. Users may also upload extensions [61] to MIT App Inventor that permit the use of more components, such as additional Bluetooth functionality.

6.2 Prior Work

The development of the PIC project was inspired by prior work in public ML tools for high school students and the general public: tools created by the PAIR and TensorFlow.js teams and another tool created at MIT App Inventor.



AboutScreen
AccentColor: Default
AlignHorizontal: Left : 1
AlignVertical: Top : 1
AppName: mole
BackgroundColor: Default
BackgroundImage: None
CloseScreenAnimation: Default
Icon: None
OpenScreenAnimation: Default
PrimaryColor: Default
PrimaryColorDark: Default
ScreenOrientation: Unspecified

Figure 6-4: An MIT App Inventor project is built using both a designer interface and a blocks interface. The Designer interface (top) allows users to drag-and-drop and customize components such as buttons onto a mockup Android screen. The Blocks interface (bottom) provides code blocks that can be arranged to form basic programs. This example shows an implementation of the "Mole Mash" tutorial from the MIT App Inventor website [58].

6.2.1 TensorFlow.js and Webcam Pacman

TensorFlow.js is a JavaScript library for training and deploying ML models in the browser or on Node.js [116]. The TensorFlow.js website provides resources for users to start working on projects, such as code demos, prebuilt models, and data visualization tools.

The TensorFlow.js team used their library to create a demo called "Webcam Pacman," [126] a browser game where players can play Pacman using a machine learning model. Players first use part of the game's interface to train an image classifier to recognize images with labels corresponding to the four directions of in-game movement. When the game starts, they can play by placing the correct objects in front of their webcam to move.

In Webcam Pacman, the model is trained using transfer learning (see Section 4.3). The images are inputted into a pre-trained MobileNet model [48]. The outputs from the last convolutional layer of this model are further passed into a smaller model to be trained. Transfer learning provides two benefits for Webcam Pacman: the model performs efficiently in a browser, and despite providing small datasets through the webcam, transfer learning gives reasonable results.

6.2.2 The What-If Tool

What-If is an analysis tool by PAIR that allows users to analyze TensorFlow machine learning models without customizing the code [136]. Given a TensorFlow model and a dataset, What-If provides an interactive visual interface for exploring inference results, specifically, data points on a grid. This grid allows the user to pick multiple dimensions to sort the data points. Additional capabilities include generating confusion matrices, displaying counterfactual pairs of data points, and allowing the user to edit individual data points' features. What-If lowers the technical barrier to analyzing model behavior. Non-programmers can use the interface to evaluate inference results and develop intuition for model performance.

6.2.3 Teachable Machine

Teachable Machine is a web app built using TensorFlow.js that allows the general public to play with image classification [118]. Users use pictures from webcams to train three built-in classes on their browser. The webcam takes continuous video and determines which class is most similar. The advantages of Teachable Machine are: (1) it requires no programming experience, and (2) it allows easy data input with minimal file formatting constraints and no preprocessing.

As mentioned in Chapter 4, the Deep Learning Practicum course used Teachable Machine to introduce k-nearest neighbors and transfer learning. In class, students experimented with the interface and were given suggestions for ways to train combinations of objects. Each suggestion asked students to intuit the rationale for certain results, and whether these results are expected or surprising. Students with no programming experience were able to address the suggestions and design variations. Playful and intuitive tools such as Teachable Machine help students self-learn and analyze concepts and behaviors of machine learning.

6.2.4 MIT App Inventor's Look Extension

MIT App Inventor recently produced "Look", an extension that allows users to incorporate ML into their mobile applications. The Look extension provides a set of blocks for users to perform image classifications using cameras on their mobile devices [57]. Look uses MobileNet, a deep neural network with high accuracy on the ImageNet dataset and relatively low memory requirements that was specifically designed to allow mobile devices to run a high-performance image classification model [48]. The decreased computational power required allows mobile devices to load models and classify images in real time. However, the MobileNet model is limited by its training dataset (ImageNet [23]) and can only classify the 1000 ImageNet labels of plants, animals, man-made objects, and geological formations. The PIC project was motivated by a desire to personalize the types of image classifications that MIT App Inventor users could run so that they could include specific classes relevant to the apps that they want to make.

6.3 The PIC Web Interface

In order to appeal to the target audience of young ML novices, the interface needs to have no coding requirement, be intuitive to use, be personalizable, and allow users to quickly train ML models. For the no coding requirement, interactive elements provide data and structure normally handled by altering code. Users can add and delete model labels and tweak model layers and parameters. Almost every aspect of the training model can be changed in the interface without writing code.

The interface is made to be more intuitive through a progress bar that tracks the model creation process. There are four steps in total: (1) training, (2) model selection, (3) testing, and (4) viewing results. Each step has a tab and task, exposing users to relevant features only. Users enter Step 1 upon visiting the interface, and advance or return to steps using "next" and "back" buttons or by clicking on steps in the progress bar. When switching between steps, all data such as labels and examples added are preserved.

Personalization is implemented by allowing users to train image classification models for any application. Text boxes for labels are provided, and no constraints are placed on data from the webcam.

Low training time and high accuracy on smaller datasets are achieved through transfer learning. Users can train a smaller model (the "target" model) using knowledge learned by a pretrained model (the "source" model). This method was adopted from Webcam Pacman: the user's webcam images are first passed into a source model; and activations from the second to last layer of the source model are used to train a target model. Large source models are limited by training time. Thus, the interface provides two options for small source models: MobileNet, as used in Webcam Pacman, and SqueezeNet, as used in the Look extension. The target model provides the final classifications for labels specified by the user and is made small by default, although the user can add layers.

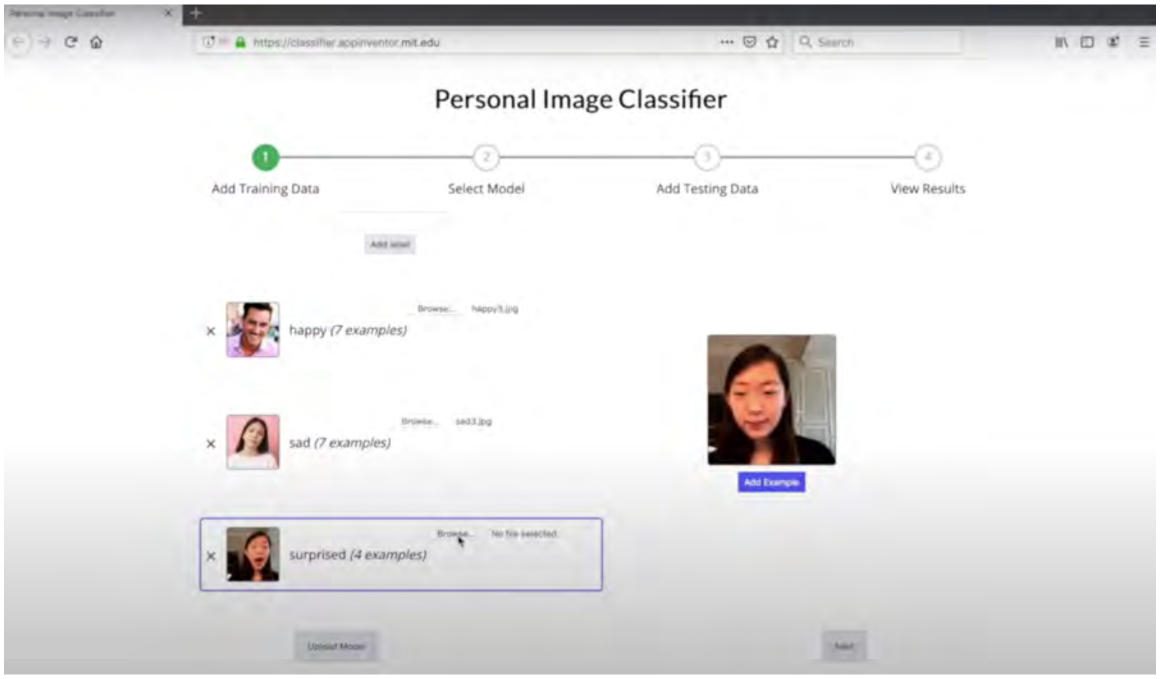


Figure 6-5: The UI for the training step for adding labels and images.

6.3.1 Programming Tools Used

The PIC web interface was written with HTML and JavaScript and stylized with CSS. TensorFlow.js handles all machine learning tasks such as training and model download. The application is run using Node.js with Yarn as the package manager. It is hosted using http-server on a Linux virtual machine provided by MIT App Inventor.

6.3.2 Step 1: Data Input

In Step 1, users add labels and provide examples using webcams. The Step 1 UI has two columns (see Figure 6-5). The left column contains information about the labels added by the user, while the right column contains a webcam feed with a button for adding examples. The user may also upload images from their computer through the "Browse" button to the right of each label.

All labels and examples are saved in the browser and deleted if their label is removed from the interface. A dedicated JavaScript file containing a class named "Dataset" handles all operations regarding adding, removing, and formatting images and labels. Two instances of this class are instantiated for the training data: one to contain activations from MobileNet and one to contain activations from SqueezeNet. The instance used depends on the source model selected in Step 2. Three main components are saved: labels, images, and activations from a source model.

When an image is inputted by the user, the following occurs:

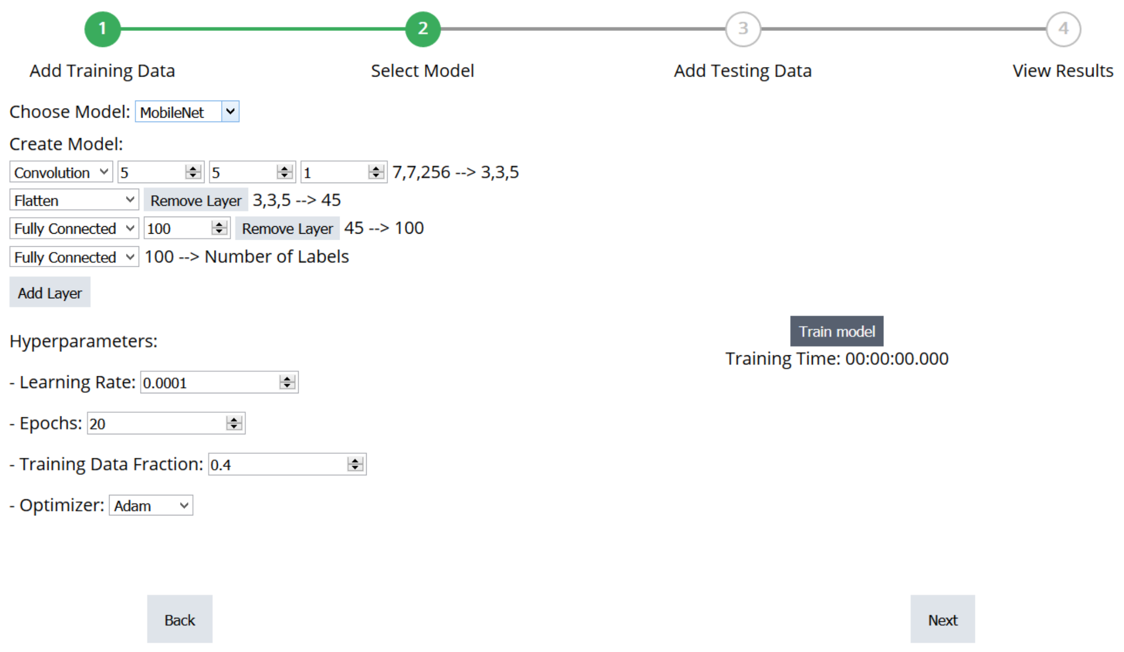
1. The image is converted to a tensor so that it can be used with TensorFlow.js.
2. The tensor is passed through MobileNet and SqueezeNet. Activations on these tensors from the models' second to last layers are kept.
3. For each source model, both the original image and its activation are saved in the model's corresponding dataset.

Model training time was optimized by saving a different training set for each source model, which did not substantially affect the webapp's memory requirements.

6.3.3 Step 2: Model Editing and Training

In Step 2, users can choose the source model and tweak the target model for the PIC transfer learning process. The UI has two columns (see Figure 6-6) for editing and training the model respectively. Beginners can accept default values provided, and bypass model editing. Training is initiated in the second column, and training time and model loss are displayed. When training is complete, users are automatically brought to Step 3 for testing.

The left column allows selection of the source model, editing of the layers of the target model, and editing of model hyperparameters. The default model has one convolution layer, one flattening layer, two fully connected layers, and ends in one softmax layer that is not shown. Layer types that can be added include fully connected, convolution, max pool, and flattening. Layer parameters, such as window size and stride, may be edited. Editable model hyperparameters include learning rate, number of epochs, fraction of the training set to use as the batch size, and optimizer function.



Choose Model: MobileNet

Create Model:

Convolution 5 5 1 7,7,256 --> 3,3,5

Flatten Remove Layer 3,3,5 --> 45

Fully Connected 100 Remove Layer 45 --> 100

Fully Connected 100 --> Number of Labels

Training Time: 00:00:00.000

Hyperparameters:

- Learning Rate: 0.0001
- Epochs: 20
- Training Data Fraction: 0.4
- Optimizer: Adam

Figure 6-6: The UI for the model editor step. Users can tweak details about their model and start the training process here.

Model editing is handled with its own JavaScript file, which contains two key classes: "LayerNode" and "LayerList". A LayerNode is a flexible representation of a single layer in the model, and stores all parameters associated with the layer. It also contains a method to create the layer's associated HTML elements to append to the interface. A LayerList is a doubly linked list that connects all the layers of the user's custom model. LayerLists allow adding and removing layers, and checking model validity and generating various invalid model messages. During training, each layer is added sequentially to a sequential model using TensorFlow.js.

During training, a method in the training dataset instance formats the dataset's images and their corresponding labels as inputs to the model training method. This method outputs an object that contains three tensors. The first tensor contains all of the dataset's images' activations. The second tensor contains all of the dataset's images' labels after they are mapped to numerical indices. A separate mapping of these indices back to the text labels is calculated and stored, so that the original labels can be retrieved from model predictions. The third tensor contains all original images from the dataset, which is used to make thumbnails for the results page in Step 4.

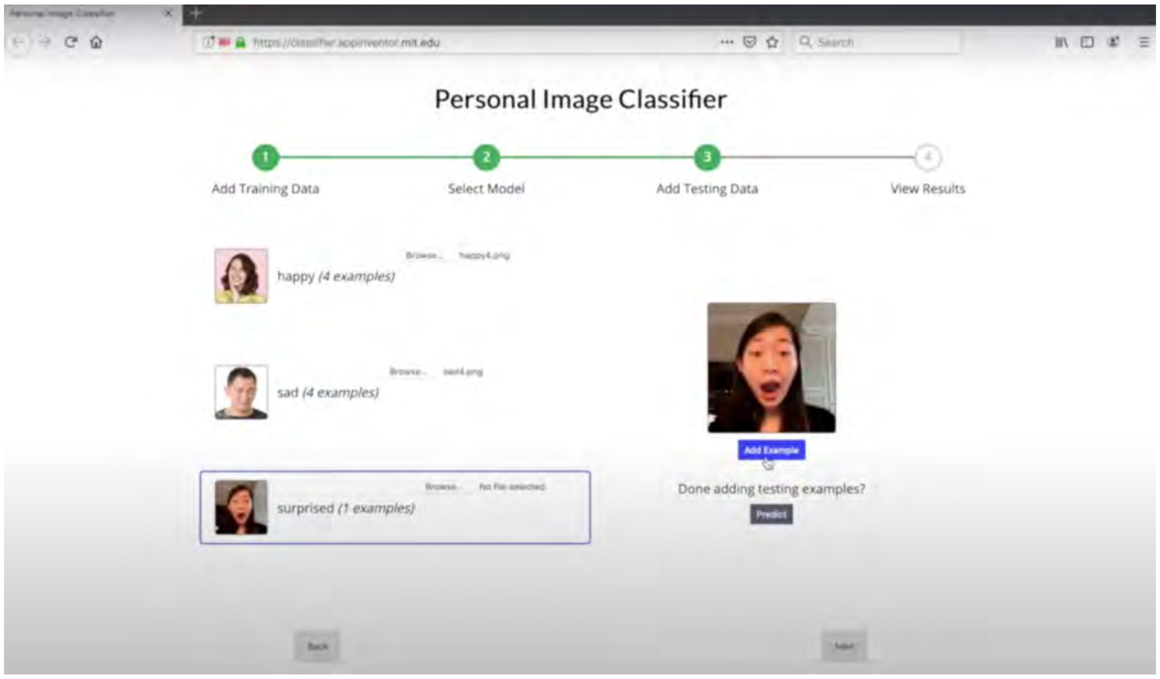
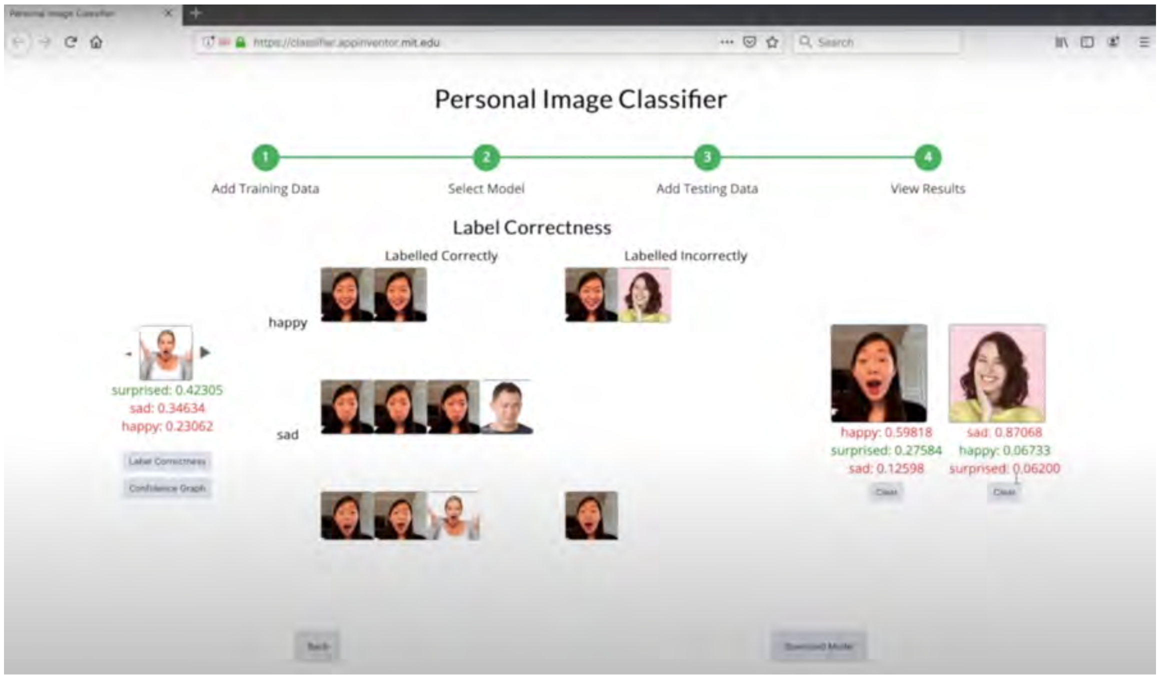


Figure 6-7: The UI for the model testing step. Users can add new images in the same way as in Step 1 to create a new testing dataset.

After these tensors are obtained, the dataset's activations and indices are passed over to TensorFlow.js's model training method, alongside the hyperparameters set by the user.

6.3.4 Step 3: Model Testing

The UI for model testing is similar to that of Step 1, but the data collected during this step is used for testing instead of training (see Figure 6-7). When the "Predict" button is pressed, the testing images are passed through the source model, and then the appropriate activations are passed into the target model to output the final predictions. A separate instance of the "Dataset" class is instantiated to handle all images added to the testing set. After prediction of labels, users are brought to Step 4.



surprised: 0.42305
sad: 0.34634
happy: 0.23062

happy: 0.59818
surprised: 0.27584
sad: 0.12598

sad: 0.87068
happy: 0.06733
surprised: 0.06200

Figure 6-8: The UI for the results step. Users can analyze their model's predictions on the testing set here.

6.3.5 Step 4: Results and Analysis

Step 4 helps users gain insight into model behavior and inform retraining. Users can view top three predictions for each testing image, select analysis tools (confidence graphs vs correctness tables), and compare images side by side (see Figure 6-8).

Formatting results is handled by a separate JavaScript file containing a class named "Results". An instance of this class is constructed when the model finishes the prediction task. This instance stores all of the data needed to display results to the user, including:

1. A mapping of model prediction indices to label names.
2. A tensor of the testing images, to use in thumbnails and in the comparison canvases.
3. A tensor of model prediction indices, where every three indices correspond to the top three predictions for each image.

4. A tensor of model prediction values, where every three values correspond to the top three predictions for each image.

5. A tensor of the images' actual labels as model indices.

Each component in the results page that displays an image thumbnail and/or prediction values relies on a method called getResult from the Results class. This method formats a single image's result data into a JavaScript object that contains the image, the name of its actual label, the names of its top three predicted labels, and its top three predicted values. There are additional helper methods to track image indices, and a method to return results for all images, to help populate the analysis tools.

Correctness Table

The default analysis tool displayed in the middle of the results page is the Correctness Table, which shows all testing images and whether the classification was correct (see Figure 6-8). If the model's top prediction matches the true label (regardless of the probability value), the classification is correct. This table helps users assess why specific images were classified correctly or not. Common points of discrepancy or similarity that can be assessed include details in object positioning, angle, lighting, color, background, and proximity to the webcam.

Confidence Graph

The other analysis tool available is the Confidence Graph (see Figure 6-9). For a given label, this graph buckets all testing images based on prediction confidence. The confidence buckets include "medium," "high," and "very high" confidences. Very high confidence includes prediction probabilities of 0.8 or higher, high confidence includes probabilities of 0.6 or higher, and medium confidence includes probabilities of 0.4 or higher. Images with lower than 0.4 confidence predictions are not displayed.

The Confidence Graph allows users to infer the characteristics of images that a model learns for specific labels, and to plan a retraining strategy. Display of the

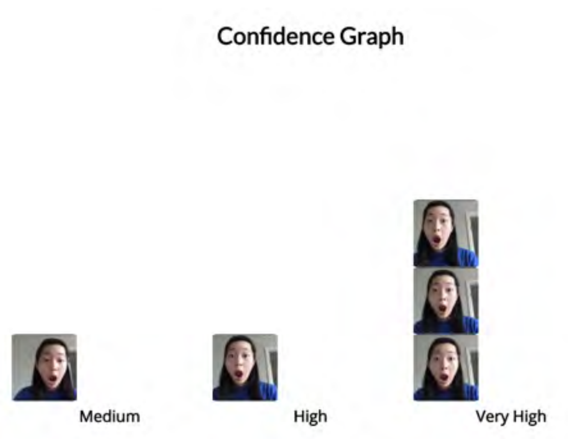


Figure 6-9: The Confidence Graph shows prediction results by label. In this screenshot, the label selected is "surprised". A user may notice that the image classified with the lowest confidence has the person facing the opposite direction as all of the other images. Not shown here is the dropdown menu at the top of the graph that lets users select which label to analyze.

analysis tools is handled by a separate JavaScript file. Each analysis tool has two methods which are called when a user clicks on one of the analysis tool buttons: one for building its HTML outline and one for populating it with data. These methods employ other helper methods that interface with the Results class to obtain the testing results. Hover and click event listeners were added to each image to power the comparison canvases.

6.3.6 Step 5: Model Download and Upload

When downloading the model, a custom download method packages all key information about the model in a zip file named "model.mdl". The zip file contains four files:

1. Two files that detail the model's topology and weight data, which are necessary for TensorFlow.js to load the model elsewhere.

2. A JSON file that maps the model's prediction indices to label names, so that MIT App Inventor can translate the model's predictions to text.

3. A JSON file with information about the source model used in the training

process, so that MIT App Inventor can load the correct one when it needs to make predictions.

Users can also upload the model file back into the interface, which adds the model's labels and allows for immediate model testing.

6.4 The PIC Extension for MIT App Inventor

The PIC extension aims to empower users to apply the models they built with the PIC webtool to create meaningful mobile applications through MIT App Inventor.

6.4.1 Back-end

Extensions to MIT App Inventor are uploaded as ".aix" files, which are built from all of the files needed for the extension to work. For the PIC extension, this includes a Java file and an assets folder. The Java file, which has a class called PersonalImageClassifier, contains the main functionality of the extension. This class is an implementation of the Component class, and has all of the code that builds the extension's properties and blocks. The assets folder carries all outside dependencies that the Java class refers to. These assets are downloaded together with the mobile application itself, and the Java class contains an annotation that points to where these assets are.

Since the PIC webtool uses TensorFlow.js to train and download models, TensorFlow.js is also used in the extension to load these models and to make predictions. The following files are added to the assets folder:

1. The TensorFlow.js library, so that it can be used locally in the mobile application.

2. Files containing all of the information needed to load MobileNet and SqueezeNet with TensorFlow.js.

3. A Javascript file containing all of the methods and logic for using TensorFlow.js to load the source models, to load the trained target model, and to make predictions.

4. An HTML file that will load the TensorFlow.js library and aforementioned Javascript file, as well as allow us to interface with the main Java class of the extension.

After these files are included, the main Java class has to be able communicate with the Javascript file. The solution to this is to include a WebViewer component as a property of the extension. Typically in MIT App Inventor applications, this component allows the user to view web pages, but it also has the ability to handle Javascript. In the extension, the component loads in the HTML file, which in turn loads in the TensorFlow.js library and Javascript file. This gives the WebViewer access to the Javascript file's methods so that they can be called from the Java file. Finally, the WebViewer component provides a client for the Javascript file to make API requests to fetch files for loading models: the Javascript sends API requests for these files, and the client receives them and returns the files in its response.

The final piece needed to make the extension work is support for reading the ".mdl" files downloaded from the PIC interface. Since this file is uploaded to MIT App Inventor as a property of the PIC extension, the Java class receives the path of where the file is located. This path is saved in a private field in the class so that the model file can be accessed multiple times. Whenever this file needs to be opened, a zip file reader from Java's utils library is employed.

When the Javascript file is first loaded by the WebViewer component, it sets up all of the TensorFlow.js models it needs using the following series of API requests:

1. A request is sent to get information about what transfer model to use. The WebViewer client responds with the JSON file containing this information from the model file.

2. Multiple requests are sent to get all parts needed to load the correct transfer

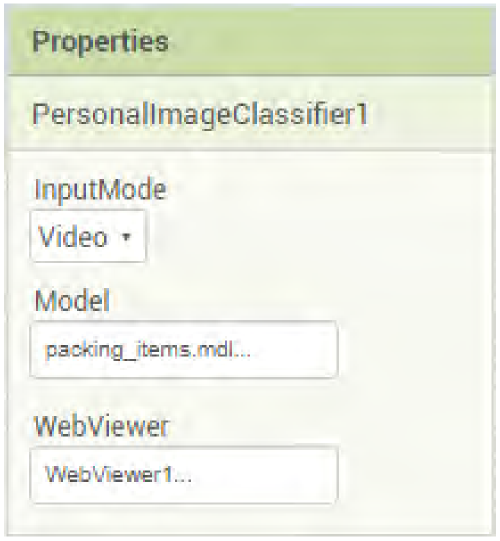
model. The WebViewer client responds with the corresponding files from the assets folder.

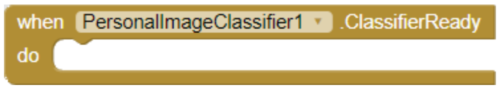
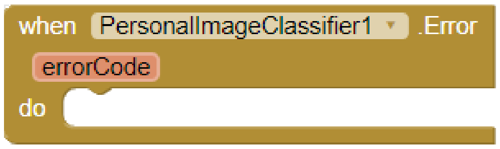
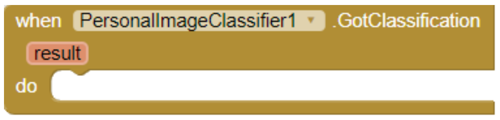
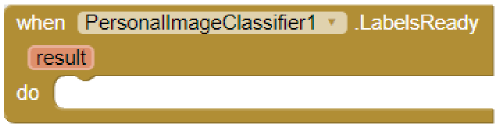
3. Two requests are sent to get the model topology and weights to load the user's trained model. The WebViewer client responds with these files from the model file.

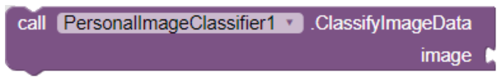
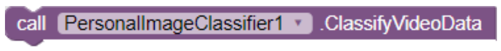
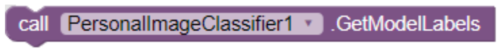
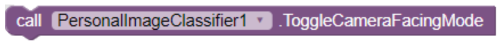
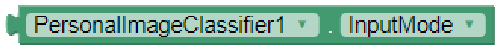
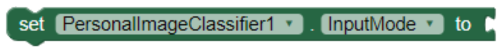
4. A request is sent to get a mapping of model outputs to label names. The WebViewer client responds with the JSON file containing this mapping from the model file.

6.4.2 Extension Properties and Blocks

The PIC extension has three properties in the designer and 11 blocks in the blocks editor. It shares many blocks with the Look extension, which was used as a starting point. All of the PIC extension's properties and blocks are outlined in Table 6.1.

	Block Image	Function
a.	 Properties PersonalImageClassifier1 InputMode Video Model packing_items.mdl... WebViewer WebViewer1...	The *InputMode* property has two settings that control if the extension takes pictures from a continuous video feed or from a single camera snapshot. The *Model* property is where users can upload and select model files from the PIC interface to use. The *WebViewer* property is where users can attach a WebViewer component to use the extension with.

b.		The *ClassifierReady* event block is triggered when the extension is finished loading the user's model and transfer model. This block is the starting point of functionality that uses this extension.
c.		The *Error* event block is triggered when the extension runs into any errors. It returns the number code that is associated with the error.
d.	 when PersonalImageClassifier1 .GotClassification result do	The *GotClassification* event block is triggered when the extension is finished performing a prediction on an image. The event is expected to be fired after either the *ClassifyImageData* or *ClassifyVideoData* blocks are run. It returns a list of lists of the top 3 prediction results.
e.		The *LabelsReady* event block is triggered when the extension is finished fetching the model's labels. The event is expected to be fired after the *GetModelLabels* block is run. It returns a list of the model's labels.

f.	 call PersonalImageClassifier1 .ClassifyImageData image	The *ClassifyImageData* block initiates the classification of an image that it takes as an input. When the classification is complete, the results are returned through the *GotClassification* event block.
g.	 call PersonalImageClassifier1 .ClassifyVideoData	The *ClassifyVideoData* block initiates the classification of the current screenshot in the camera's video feed, which may be seen using the WebViewer component. When the classification is complete, the results are returned through the *GotClassification* event block.
h.	 call PersonalImageClassifier1 .GetModelLabels	The *GetModelLabels* block initiates the fetching of the uploaded model's labels. Results are returned through the *LabelsReady* event block.
i.	 call PersonalImageClassifier1 .ToggleCameraFacingMode	The *ToggleCameraFacingMode* block switches the user's camera mode between selfie and outward facing.
j.	 PersonalImageClassifier1 . InputMode	The *InputMode* getter block returns the current setting of the extension's *InputMode* property.
k.	 set PersonalImageClassifier1 . InputMode to	The *InputMode* setter block allows users to set the extension's *InputMode* property. There are two options: "Video" and "Image".

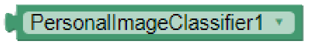
l.		The *PersonalImageClassifier* getter block returns a specific PersonalImageClassifier extension instance.

Table 6.1: An overview of the PIC extension's properties and blocks.

6.5 Usability Workshops with High School Students

To test the usability and effectiveness of the PIC webtool and extension for engaging novices in ML, we designed and conducted workshops for high school students. The workshops were run with two classes of AP Computer Science Principles (CSP) [15] students from Boston Latin Academy on March 12, 2019 and March 15, 2019. Each workshop lasted for two class periods, each about 50 minutes long. These students were inexperienced in ML but had prior experience using MIT App Inventor through the AP CSP class.

The workshops involved creating a mobile game that awards users with points for making specific facial expressions. The game detects facial expressions using an image classifier that the user trains on their face. Points are awarded based on whether or not the prediction matched the goal expression and how high the output probability is. The lesson plan, informational video, teacher guide, slides, and tutorial worksheets created for these workshops are accessible for free on the MIT App Inventor website [60].

The first workshop introduced machine learning and involved using the PIC interface to build a model for recognizing various facial expressions (see Table 6.2). The workshop began with a short introduction to basic concepts of ML. Then, students split up into groups of 4-5 to build image classifiers for recognizing facial expressions of their choice. Instructors were available to answer questions and provide suggestions.

The second workshop involved having students use the PIC extension in MIT App Inventor and the models they trained (see Table 6.3). The students were provided the

Time	Activity
7 min	Introductions, explanation of MIT App Inventor and purpose of study, pre-questionnaire.
10 min	Introduce CSP students to Machine Learning. What is it? What does it mean? • Compare it to a function (a "box" that receives inputs and outputs answers), but is different in that there may not be a clear answer for machine learning tasks. • Give examples (speech recognition, text generation, image classification). • Introduce how machine learning models get better at their tasks by repeatedly training and testing. Give basic tips on how to approach this such as adding many varied examples.
33 min	Students split into groups to use the PIC interface to train a model to recognize their facial expressions. A tutorial on how to use the interface was provided. Make sure that they download their models at the end of class.

Table 6.2: First PIC workshop lesson plan.

mobile application file to upload to MIT App Inventor, and uploaded their models to the application using the extension. Instructors were again available for questions and suggestions.

6.5.1 Instruments

Written questionnaires were collected before the first workshop and after the second workshop. Questions fit into three categories: understanding of machine learning concepts, understanding of ML applications, and effectiveness of tools used. The pre-questionnaire had 16 questions and gauged students' prior experience in and understanding of machine learning. Topics included prior exposure to machine learning, common ML applications students saw or used, feelings toward machine learning, and the most important parts of a machine learning project.

The post-questionnaire had 29 questions and focused on all three categories of

Time	Activity
25 min	Students have more time to work in their groups to create image classifiers. Ask them to use what they learned from the first workshop to create new models that perform better.
15 min	Use the PIC extension for MIT App Inventor. • Download the application file and upload it to MIT App Inventor. • Upload the model using the PIC extension. • Play the game on a mobile device. • Iterate over the model again if necessary.
10 min	Have students fill out the post-questionnaire.

Table 6.3: Second PIC workshop lesson plan.

questions. It repeated most questions from the pre-questionnaire, and added questions about students' experiences with the PIC tools and during the workshop. Topics included the ease of using the PIC interface, interesting results from using the analysis tools, and student confidence in discussing their project with others. The post-questionnaire sought to both measure what the students learned from the workshops, and the effect of PIC tools on learning.

6.5.2 Results

The workshops were run with two classes of high school students, who ranged between 10th and 12th grade. The two classes had 26 and 28 students, respectively. We obtained consent for research from 15 students from the first class and 8 from the second class.

Prior Exposure to Machine Learning

When asked about prior exposure to machine learning on the pre-questionnaire, almost all of the students were aware of past exposure to ML regardless of knowledge of ML.

Question	Mean	Range
How confident do you feel about explaining/discussing ML with a non-technical person?	0.77	0-2
How confident do you feel about explaining/discussing ML with an expert?	0.32	0-1

Table 6.4: Pre-questionnaire means and ranges for students' responses to how comfortable they would be discussing machine learning with others.

Explanation of Machine Learning

To evaluate students' understanding of machine learning pre- and post- our workshops, we asked students how they would explain ML to an ML-naive friend. On the pre-questionnaire, most of the students either did not know what machine learning was at all, or had unclear ideas as to what it was.

On the post-questionnaire, there was a clear overall improvement in understanding among the students. The students' responses had greater depth that revealed knowledge about ML processes and data in particular (e.g. "*giving data to a machine in order for it to learn and automate a process*" and "*technology learning to recognize or perform certain tasks based on patterns/info that you give them.*"). Furthermore, many students incorporated their projects into their explanations. These results showed that students understood that ML systems learn patterns from human generated data, and that students could see themselves as generators of data.

Comfort With Discussing Machine Learning

Students rated their own levels of comfort with discussing machine learning with both non-technical people and with experts on a 0-3 Likert scale, where 0 = No confidence and 3 = High confidence. In the pre-questionnaire, students were not confident in their abilities to converse about machine learning with others (see Table 6.4).

In the post-questionnaire, students' confidence improved (see Table 6.5). No student felt that they had no confidence in talking to a non-technical person about ML, and 3 students felt like they could talk to experts with high confidence. These results

Question	Mean	Range
How confident do you feel about explaining/discussing ML with a non-technical person?	1.86	1-3
How confident do you feel about explaining/discussing ML with an expert?	1.41	0-3

Table 6.5: Post-questionnaire means and ranges for students' responses to how comfortable they would be discussing machine learning with others.

Question	Mean	Range
How much do you agree: I enjoyed using the Personal Image Classifier web tool	2.35	2-3
How much do you agree: The Expressions Match app was fun to use	2.57	2-3
How much do you agree: The Personal Image Classifier web tool was confusing	1.04	0-2

Table 6.6: Post-questionnaire means and ranges for students' responses to questions asking about their experiences using the tools we built.

show that the workshops are capable of empowering students to engage in machine learning tasks and discussions.

Effectiveness of Tools

During the workshop, students spent almost all of their time using either the PIC interface or playing the mobile game containing the PIC extension. The post-questionnaire demonstrated that all of the students enjoyed using the tools. When asked about the most fun part of the workshop, 17 of the responses were about taking pictures of their own facial expressions and 3 of the responses were about playing the game at the end. Almost all students did not find the PIC interface to be confusing to use (see Table 6.6). These results suggest that the interface is likely usable and intuitive for beginners with no prior machine learning experience.

When asked how they used the analysis tools, students responded that they were either able to analyze their results or use them to further improve their models. Some students were able to use the analysis tools to develop reasoning for why their models were behaving in certain ways (e.g. "*We kept getting low scores. Then when we zoomed in to take pictures, our scores were much higher. That's because they were zoomed in during the model*"). Other students were able to recognize strange or interesting behaviors of their models through the analysis tools, even if they could not act upon those behaviors ("*it recognized people making different shirt colors and expressions as the same people*"). Overall, these results suggested that students were able to develop intuition into their ML models through the analysis tools.

6.6 Discussion

PIC was well-received in the MIT App Inventor community after its release, and has been included in many K-12 AI education collections by various institutions and organizations [107, 40, 140, 4]. Notably, PIC is being used internationally by the National Taiwan Normal University in elementary and junior high AI curricula. Researchers created the "AI 2 Robot City" curricula, which combines PIC with MIT App Inventor robotics components to make a game-based "instructional tool and model integrating the personal image classifier of MIT App Inventor with a computational thinking board game named 'Robot City'." [50]. Teacher training sessions have continued through the COVID-19 pandemic to help teachers support student learning of AI through online education (see Figure 6-10).

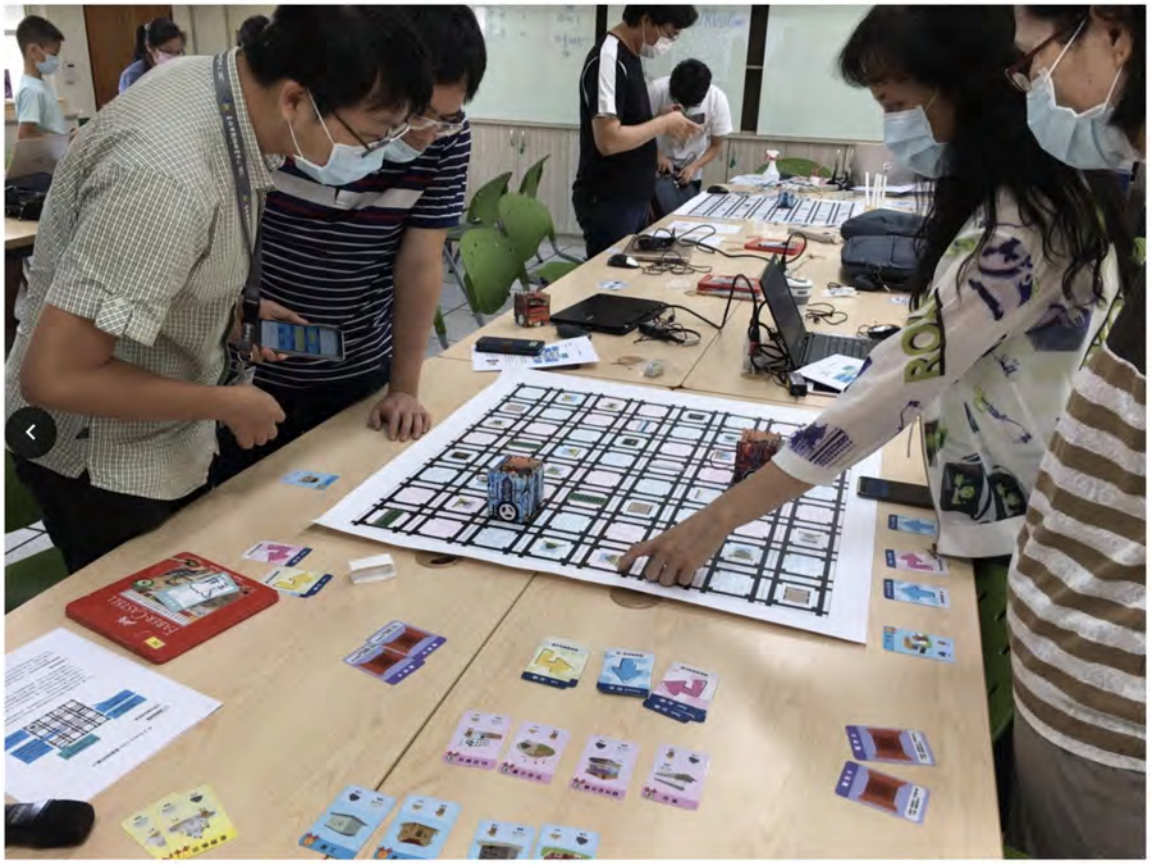


Figure 6-10: PIC teacher training at the National Taiwan Normal University [50]. The PIC webtool and extension are used in the "AI 2 Robot City" curriculum for K-12 to classify cards that guide a robot car through a simulated city.

7

The PAC Tool: Personal Audio Classifier

In this chapter, I present the Personal Audio Classifier (PAC), a public ML tool similar to PIC for high school and above students. PAC allows users to train and test their own audio classification models, and apply these personalized models in mobile applications built using MIT App Inventor. PAC was built with the same motivations and overarching design principles as PIC. I will discuss the technical challenges of PAC that differed from PIC and the improvements made to the user interface. PAC is hosted at https://c1.appinventor.mit.edu/. Most of the material in this chapter has been previously published [12, 11].

7.1 Differences and Design Improvements from PIC

Due to PIC's popularity in MIT App Inventor, many users requested a similar tool for classifying different types of sounds. The PAC webtool obtains short audio clips from a computer's microphone, converts the audio data into image representations of the spectrum of frequencies and amplitudes as they vary with time (spectrograms), and then uses transfer learning to train a spectrogram classification model in a similar way to PIC. Users may then upload these models into the PAC MIT App Inventor extension to use within their mobile applications.

We incorporated feedback from PIC users to improve on the design and flow of PAC. Instead of giving users the ability to select a source model, architect the target model, and set hyperparameters, Step 2 of PAC only involved tuning the hyperparameters of a preset transfer learning model. We found that only the hyperparameters had much impact on the model's final performance and that the model architecture portion made many users feel confused or overwhelmed, especially since most middle school and high school level curricula do not go into the details of how each layer of a neural network works.

We also condensed Step 3 and 4 from PIC (3: Model Testing and 4: Results and Analysis) into one step in PAC so that users could input new test examples and see the results live. We modified the analysis tool from Step 4 of PIC to just a simple correctness table in PAC, since the confidence graph was not as useful to students.

Finally, we made extra efforts to make PAC more colorful and have a sharper looking UI: public ML tools for K-12 should be visually appealing to grab the attention of students and teachers, but PIC was not very visually appealing to many students.

7.2 Theoretical Considerations of Spectrogram Audio Classification through Image Classification

PAC uses spectrogram image representations of audio. Spectrograms allow visualization of frequency, time, and amplitude, by computing the Fast Fourier Transform over short overlapping sample windows across the audio file. An example spectrogram of a three-second audio clip with a sample rate of 44,100 Hz is shown in Figure 7-1 [45]. In the past 3 years, Khamparia and Boddapati have shown that existing image classification techniques can achieve impressive results on spectrogram audio classification tasks with very few changes to typical convolutional model architectures [65, 13]. These results provide the basis for building an in-browser audio classifier using spectrograms as the data representation.

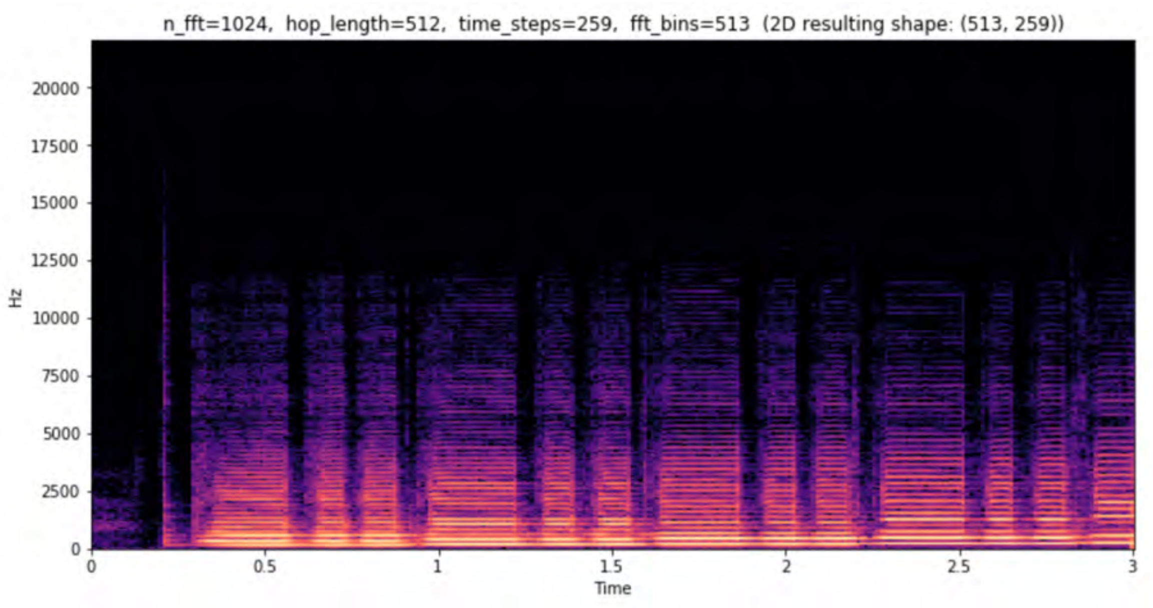


Figure 7-1: The final spectrogram output depicting frequency, amplitude, and time over a three second saxophone clip [45].

7.3 The PAC Web Interface

While many existing in-browser tools have targeted the widely popular domain of image classification, less work has been done to make the field of audio classification more accessible to students and ML novices. PAC was built to offer a simple and accessible tool to empower those without significant ML knowledge to understand and utilize audio classification in their own research, projects, or applications. The PAC front-end is built primarily in React, with custom components written in JavaScript. The front-end server is hosted via Express and Node. The spectrogram back-end is written in Python, and the back-end server is hosted via Gunicorn and Flask. Both servers are hosted on a Linux virtual machine owned by MIT CSAIL.

7.3.1 Step 1: Data Input

On the initial landing page for PAC, users begin building a custom audio classifier. The UI allows users to record audio clips and add classification labels. Each label corresponds to an instance of a "Label" React component, with its user-specified name, and an initially empty dataset of training images (see Figure 7-2). Each label

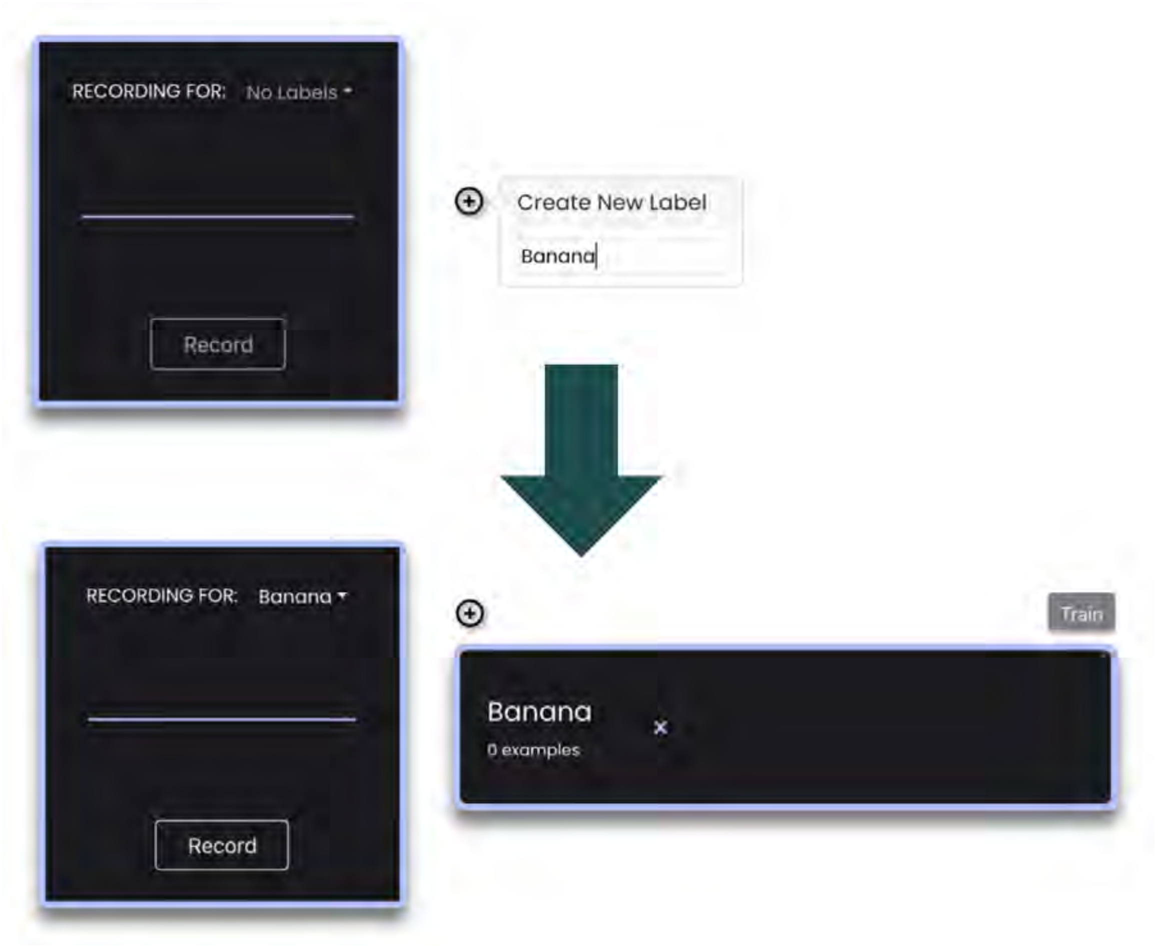


Figure 7-2: The initial PAC web interface provides a recording interface and a button to add custom classification labels. Upon entering a label name, a label is added to the right column of the PAC UI. Each label is associated with an initial empty dataset of training images.

has a number of functions that allow for spectrograms to be added to and removed from the dataset.

During recording, a custom React library opens the device microphone for one second, recording the device's input audio as a WAV file. The WAV audio file is passed to the PAC back-end via an API call, where the file is converted to a spectrogram image. The spectrogram PNG is then returned to the front-end and stored in memory in the label's training dataset via a temporary data url. Finally, a preview of the audio spectrogram appears in the label's UI component and the indicator for the number of training samples is updated (see Figure 7-3). After recording at least two audio clips for each classification label, users can begin training the model using the model tuning UI.

The PAC back-end is responsible for receiving WAV audio files through a Flask API, performing a variety of audio modifications, and creating a spectrogram image from the audio recording. It is accessible via a Python Flask API served from a Linux virtual machine running on an MIT CSAIL server. The server listens via a "/spectrogram" POST endpoint on the 5000 port. This endpoint receives requests from the frontend, unpacks the request body into a local WAV file, and calls a helper function to perform the audio modification and spectrogram conversion.

After the request body is unpacked into a local WAV file, the back-end extracts the left audio channel from the stereo signal. Then, a helper function removes beginning or ending silence in the audio clip by iterating over all audio chunks in the recording and removing the first and last continuous audio segments that register below -20 decibels. This feature was added after extensive user testing showed that user input recordings were rarely centered in the recorded audio clip. If a user were to record two identical word samples, the resulting spectrograms might look different if the user were to start speaking the word during different segments of the recording. By removing any leading and lagging silence in the input recording, we ensure that the user's input audio is centered in the recording.

Next, the back-end splits the audio recording into 10ms chunks and removes any remaining chunks that register audio below -50dB. User testing showed that this feature made PAC models far more flexible and accurate across a variety of use cases. The intuition for this is that various users might record the same phrase, word, or number, but differences in pronunciation speed might lead to small differences in the spectrogram output. By removing naturally occurring silence in input audio clips, we can better standardize recordings to contain only audio content that will reflect meaningful frequency information in our spectrogram outputs. It is important to note that we apply a more strict silence requirement for chunks in the middle of the audio recording to ensure that important audio content is not significantly altered.

Lastly, the processed audio recording is passed through a spectrogram conversion

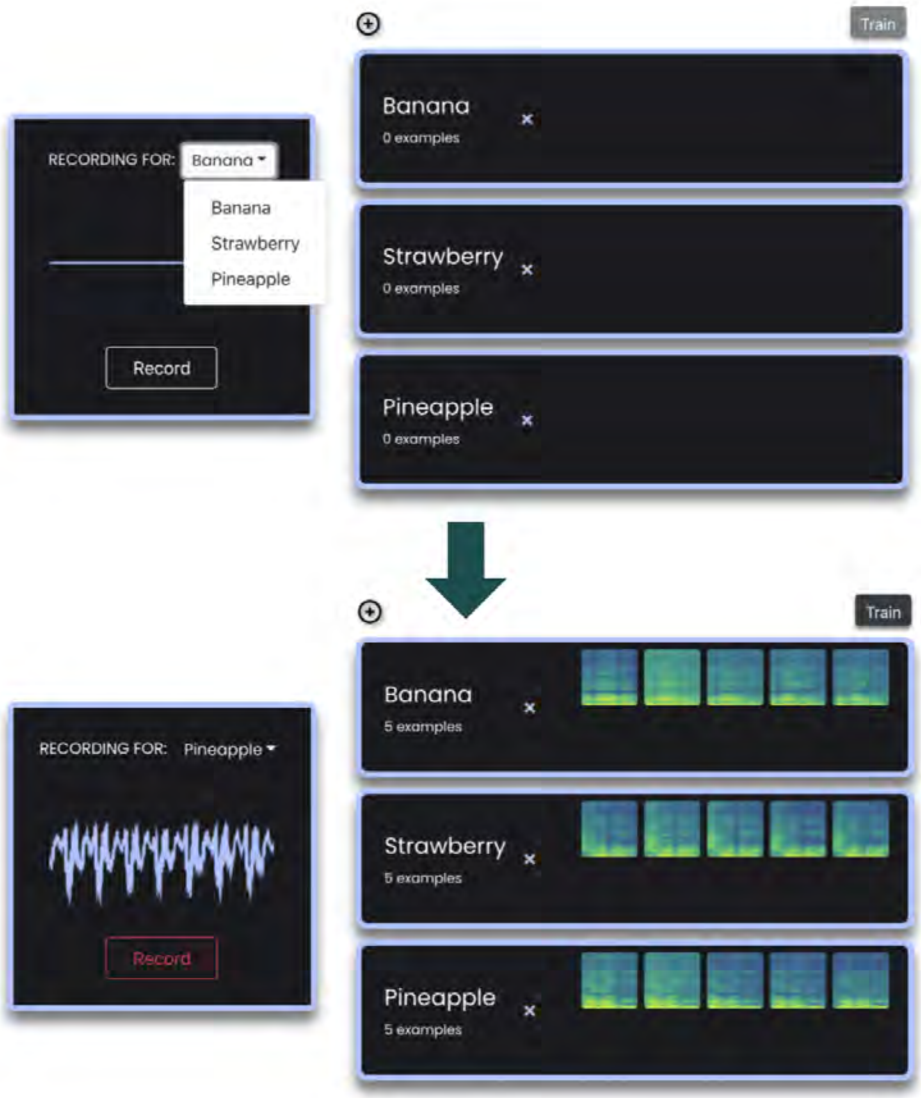


Figure 7-3: When multiple classification labels have been entered, users can select a label from the dropdown for which the recording component will record audio clips. When the "Record" button is clicked, the waveform visualizes the microphone's input and records the next one second of audio. Audio clips are passed to the backend, returned as spectrogram images, and added to the corresponding label training dataset.

function. This function splits the audio data into segments of length equal to the audio recording's sample rate, and computes the frequency spectrum for each section. A windowing function is then applied to each audio segment to connect the audio segments into a time dependent color-map.

7.3.2 Step 2: Model Tuning and Training

In Step 2, the user is able to modify the hyperparameters of the transfer learning model that will be trained to distinguish between user-specified audio classes (see Figure 7-4). PAC allows users to tweak their model in the following ways:

1. Learning Rate: a hyperparameter that controls how much the weights of the neural network are adjusted with respect to the loss gradient.

2. Optimizer: a hyperparameter that ties together the loss function and model parameters by updating the model in response to the output of the loss function in a specific way.

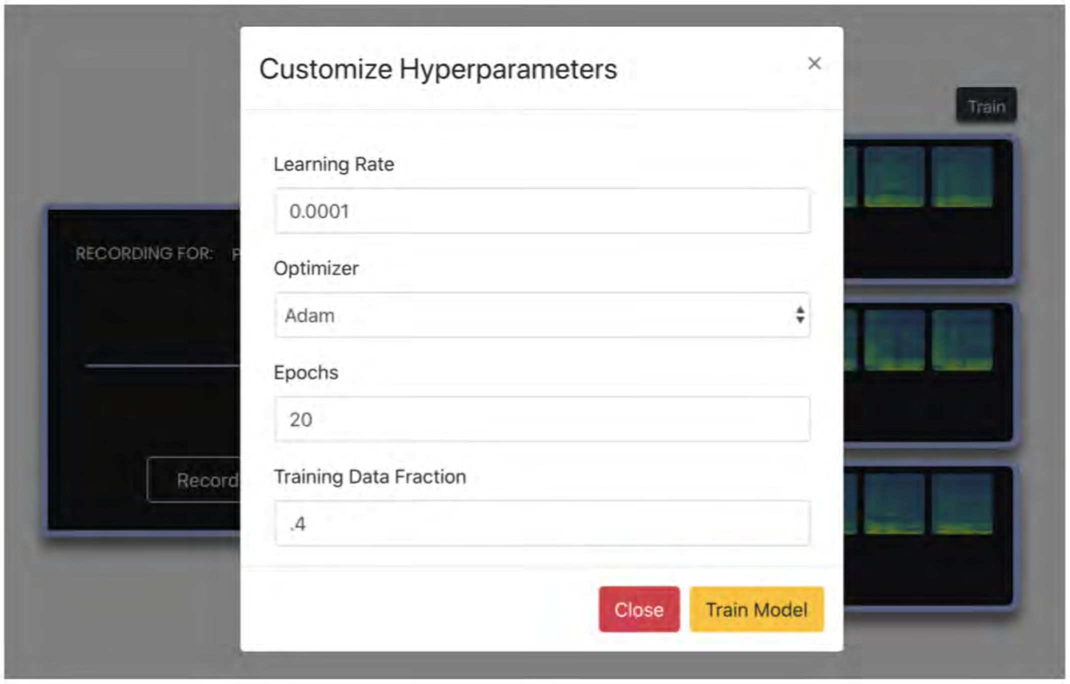
3. Epochs: the number of complete passes made through the training dataset.

4. Training Data Fraction: the fraction of the training dataset to use as the batch size.

These four hyperparameters were informed through experimentation with training the PAC model and through feedback from students who worked heavily with transfer learning models through PIC or the Deep Learning Practicum course.

After users customize hyperparameters and press "Train Model," the training datasets are passed through a pre-trained MobileNet model. Then, the output is used to train a custom five-layer image classification model with the user-specified hyperparameters. The steps for this process are as follows:

1. Each spectrogram in each label dataset is retrieved via data url and converted to a tensor for use in TensorFlow.js.


Customize Hyperparameters

Learning Rate: 0.0001

Optimizer: Adam

Epochs: 20

Training Data Fraction: .4

Figure 7-4: The model customization popup. Users can modify hyperparameters for their custom model as they see fit.

2. The tensor is passed through a modified, pre-trained MobileNet neural network, and the image tensors are replaced by the corresponding activations from MobileNet's second to last layer.

3. A custom five-layer model is trained to the specifications supplied by the user, with the MobileNet activations as the input. This model is composed of a single convolutional layer, followed by a flatten layer, two fully connected layers, and a single softmax layer. This architecture is designed to output a standard tensor with probabilities distributed across the various user-specified labels, where an output probability corresponds to the probability that the input spectrogram belongs to that label.

While the model is being trained, the PAC UI shows an interim training screen with continuously updating model loss values (see Figure 7-5). After the training is completed, users are immediately brought to the testing screen.

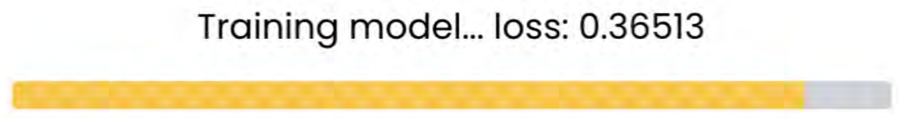


Figure 7-5: An interim loading screen shows training progress as well as the current model loss.

7.3.3 Step 3: Testing

In Step 3, users can record new audio clips for testing and see how their trained model classifies these new recordings. The UI shows the functionality from Step 1 for recording clips on the left, visualizes the input spectrogram in the middle, and lists the user-specified labels on the right. After the front-end receives the corresponding audio spectrogram from the back-end, the following steps occur:

1. The spectrogram is passed through the same pre-trained MobileNet model used in the training process.

2. The resulting activations are passed through the trained user model; the softmax layer outputs an array of probabilities that correspond to the array of labels, such that the probabilities sum to one.

3. The output probabilities are assigned to the labels and the label with highest probability is chosen as the "winning" classification.

The UI proceeds to highlight the "winning" label in green, with all other labels highlighted in red. Users can hover over each label to view the model's corresponding output probability (see Figure 7-6).

Scrolling down below the main view brings the user to the "Test Results" view, where users can visualize the results of their test audio clips (see Figure 7-7). In this view, spectrograms inputted during the testing phase are organized by how they were classified. Their corresponding confidence is shown under each image. Users can use these results to gain a better understanding of the features in spectrograms that might lead to correct and incorrect classifications, and distinguish between higher and

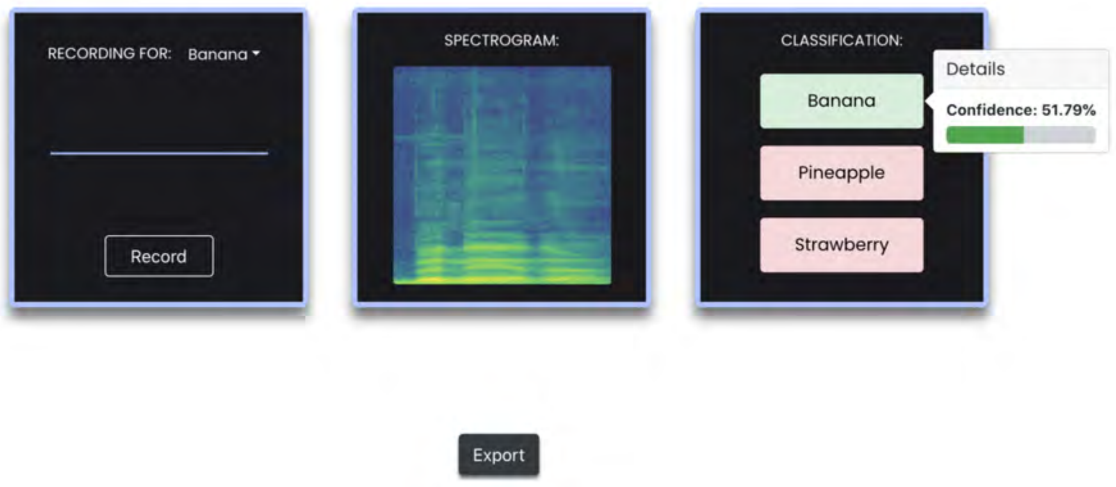


Figure 7-6: The initial testing view is made up of a recording component, a spectrogram visualizer component, and a label view component. Users record audio clips that are then passed through the trained model. The model outputs probabilities associated with each label and highlights the "winning" classification in green. Users can hover over labels to see classification confidences.

lower confidence spectrogram classifications. Users have the ability to return to the training screen at any time via the navigation bar in order to improve their model, either by removing buggy training examples, or by adding new training examples based on the insights they glean from the test results.

7.3.4 Step 4: Model Download

Similar to PIC, the web interface for PAC also has an "Export" button on the testing screen that allows the user to download the model they have trained. The ".mdl" file contains three files:

1. A model topology file detailing the architecture of the pre-trained MobileNet model.
2. A weights file detailing the weights of the pre-trained MobileNet model.
3. A JSON file mapping the model's predictions indices to their corresponding label names.

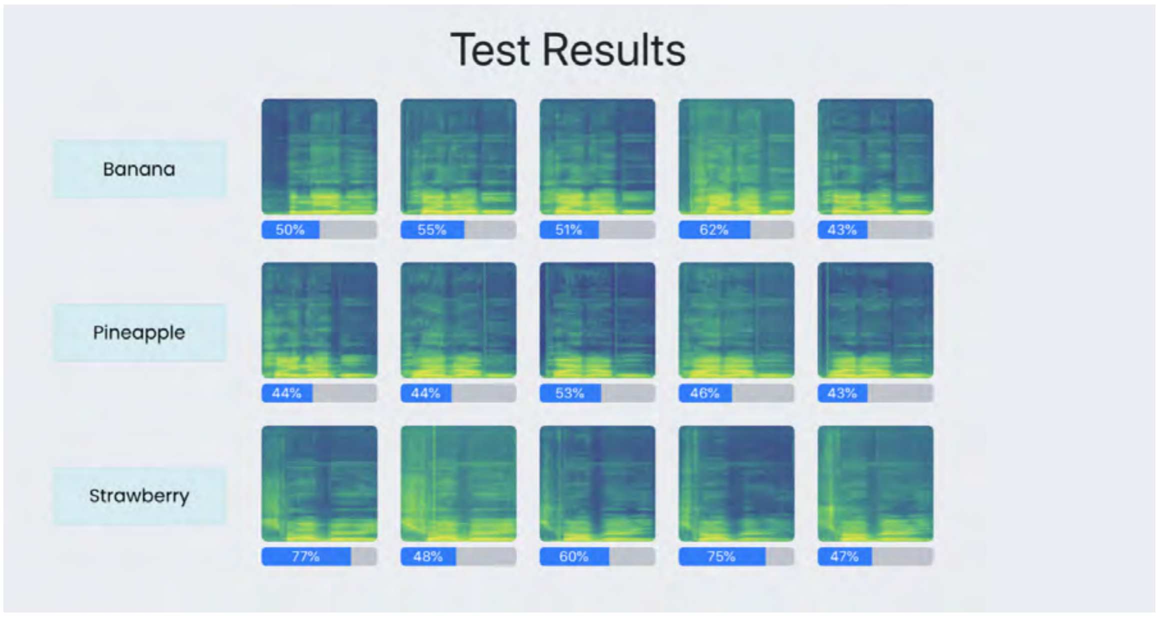


Figure 7-7: The PAC test results interface. Test audio clips are organized by their corresponding classification.

7.4 The PAC Extension for MIT App Inventor

The PAC extension aims to empower users to apply the models they built with the PAC webtool to create meaningful mobile applications through MIT App Inventor. It was built similar to the PAC extension but uses the WebViewer component for audio input in order to decrease the number of blocks required. The WebViewer component assigned to the PAC extension is configured such that a user may tap it to record an input audio clip. The PAC extension has two properties in the designer and 4 blocks in the blocks editor (see Table 7.1).

	Block Image	Function

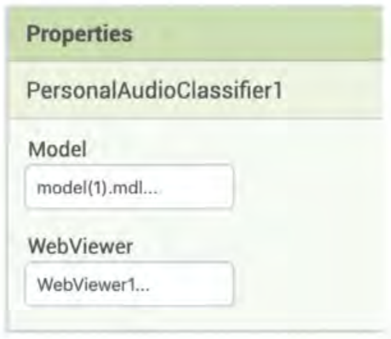
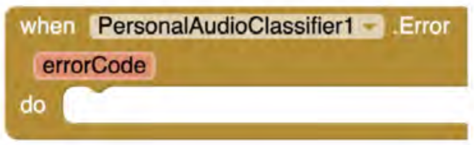
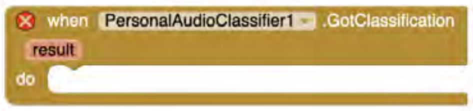
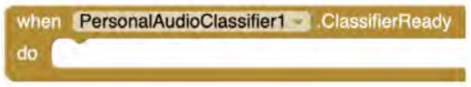
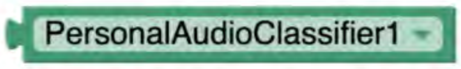
a.	Properties PersonalAudioClassifier1 Model model(1).mdl... WebViewer WebViewer1...	The *Model* designer property allows users to upload and select a model exported from the PAC interface to use in making predictions. The *WebViewer* property allows users to specify a WebViewer component to use with the extension.
b.		The *Error* block triggers when an error is encountered in the extension, and returns the code associated with that error.
c.	 when PersonalAudioClassifier1 .GotClassification result do	The *GotClassification* block is triggered after an audio clip is recorded and passed through the PAC model. The resulting classification output (the *sound* parameter) is returned in a list of lists with the top three prediction results.
d.		The *ClassifierReady* block is triggered when the extension has finished loading the MobileNet and PAC models.
e.		The *PersonalAudioClassifier* block returns a specific extension instance.

Table 7.1: An overview of the PAC extension's properties and blocks.

7.5 Usability Workshops with High School Students

To evaluate the effectiveness of the PAC webtool and the PAC extension as a learning tool for machine learning novices, we ran workshops for classes of students with limited ML experience. We designed and ran six workshop sessions with three AP Computer Science Principles (CSP) students at the Boston Latin Academy. Each class period lasted 55 minutes, and each student engaged with the workshop material across two sessions. The Boston Latin Academy classes provided a perfect workshop environment to test PAC and its related tools as students had prior experience using App Inventor tools, but had limited exposure to machine learning. The curriculum was designed around PAC and the PAC extension and focused on conveying important concepts related to artificial intelligence, data representations, and image classification. Hands-on activities were designed to engage students with in-browser neural networks and discussion topics were chosen to encourage students to think about the real-life implications of ML.

Data from the workshops was gathered via a pre- and a post- questionnaire, which were nearly identical to the questionnaires from the PIC study. These questions provided insight into student understanding of basic ML concepts, awareness of ML applications in the real world, and feelings toward the PAC webtool and PAC extension for MIT App Inventor. Each of the three AP CSP classes participated in the study over two workshop sessions. The student guide created for these workshops is accessible for free on the MIT App Inventor website [59].

In the first workshop session, the primary goal is to introduce students to fundamental machine learning ideas, and encourage them to think critically about the implications of machine learning in their everyday lives. Students start by discussing the features of an intelligent being, and agree as a class on a definition for artificial intelligence. Students then test this definition and discuss whether various pieces of technology fall under the classification of artificially intelligent. The first workshop also introduces the first in-browser machine learning tool, Google Quick Draw, and gives students the opportunity to explore doodle classification and learn more about

Time	Activity
10 min	Introductions, explanation of MIT App Inventor and purpose of study, pre-questionnaire.
10 min	Introduce CSP students to Artificial Intelligence. What is it? What does it mean? Exercises: • Explore what it means to be artificial. • Deciding what makes a human, animal, or object intelligent. • Defining artificial intelligence as a class.
15 min	Explore the ubiquity of machine learning tools in our everyday lives. Exercises: • Brainstorm examples of AI in our lives. • Segment examples by understanding vs. learning vs. planning. • Play an "Is this AI?" game as a class.
20 min	Interactive introduction to practical machine learning using Google Quick Draw. Exercises: • Explore the interactive Quick Draw game. • Discuss how Quick Draw uses data and AI. • Discuss sources of bias in Quick Draw and how they manifest.

Table 7.2: First PAC workshop lesson plan.

how models utilize large data-sets to recognize important patterns. Workshop exercises are accompanied by class discussions, slideshows, and small breakout activities. The first workshop lesson plan can be found in Table 7.2.

In the second workshop, we recap the lessons taken from Google Quick Draw, and pivot to understanding the building blocks for image and audio classification. Students learn the basic inputs and outputs of machine learning models, and understand the differences between training and testing. Much of the second workshop focuses on students exploring PAC and using what they've learned to build effective audio classification models. After exploring three unique types of PAC models, students export their models and use the PAC extension to plug these models into a pre-built Voice Authentication Diary mobile app. Students configure the pre-built app using MIT App Inventor blocks and their own PAC models to "open" for the user's voice and not their classmates' voices. The second workshop concludes with a discussion of students' successes and failures during the workshop, as well as demonstrations of students' applications. The second workshop lesson plan can be found in Table 7.3.

7.5.1 Results

The workshop sessions were run with three classes of high school students. Due to changes in research policies from prior years, MIT App Inventor and the Boston Latin Academy Research Team reached an agreement that allowed us to collect and report anonymized data for all 18+ students in the AP CSP classes. All personalized data was discarded after the study's completion. We ran the workshop for 80 students across the three classes. While questionnaire responses were collected from all 80 students, only the responses and data collected from the 28 students who were 18+ are analyzed and reported. The remaining 52 students' responses were used to provide general sentiments around the workshop and teaching success.

Prior Exposure to Machine Learning

In the pre-questionnaire, we asked about students' prior exposure to ML. As expected from prior studies with PIC in this setting, the majority of students had heard of machine learning, but fewer than a third of students were confident that they could define it. Students were aware that machine learning has permeated their everyday lives, but had a limited understanding of how machine learning worked and its relevance to new technology.

Time	Activity
10 min	Understand the building blocks of machine learning. How exactly does it work? Exercises: • Understand the inputs and outputs in machine learning. • Use Quick Draw to discuss training and testing. • Introduce other relevant types of learning: image classification, audio classification.
20 min	Explore audio classification as a learning tool through the PAC webtool. Exercises: • Number classification: build a model that can distinguish between students speaking three different numbers. • Free classification: explore PAC, build a unique model that can distinguish between a variety of words or sounds. • Voice classification: students pair up, train a model to recognize each student's voice.
15 min	Build and explore the Voice Diary App Inventor application. Exercises: • Export PAC voice classification model for use in the Voice Diary app. • Download the Voice Diary app, set up the PAC extension using a custom PAC model. • Test model and application functionality, add custom block logic.
10 min	Concluding remarks, questions, post-questionnaire.

Table 7.3: Second PAC workshop lesson plan.

Question	Mean	Range
How confident do you feel about explaining or discussing machine learning with a non-technical person?	0.55	0-3
How confident do you feel about explaining or discussing machine learning with a machine learning expert?	0.23	0-2

Table 7.4: Pre-questionnaire means and ranges for students' responses to how comfortable they would be discussing machine learning with others.

Explanation of Machine Learning

To evaluate students' understanding of machine learning pre- and post- our workshops, we asked students how they would explain ML to an ML-naive friend. In the pre-questionnaire, answers were generally vague and primarily described machine learning as a method of emulating human intelligence. The majority of students demonstrated a limited understanding of machine learning and how it worked.

The post-questionnaire posed the same question, with significantly better results. Responses commonly cited specific components of the ML pipeline, including classification, training, and testing, which reflected an improved understanding of what machine learning is and how it works. Every student who either couldn't define machine learning or defined it in terms of emulating human intelligence was able to generally describe the process by which models use data to create representations and learn from patterns.

Comfort with Discussing Machine Learning

Students rated their own levels of comfort with discussing machine learning with both non-technical people and with experts on a 0-3 Likert scale, where 0 = No confidence and 3 = High confidence. In the pre-questionnaire, student responses leaned heavily towards 0 for both questions (see Table 7.4). These results also reflected the students' written responses regarding their understanding of ML; a few students had some idea of what machine learning is, but most had only a cursory understanding of the concept and would not be comfortable discussing it with others.

Question	Mean	Range
How confident do you feel about explaining or discussing machine learning with a non-technical person?	2	1-3
How confident do you feel about explaining or discussing machine learning with a machine learning expert?	1.25	0-3

Table 7.5: Post-questionnaire means and ranges for students' responses to how comfortable they would be discussing machine learning with others.

After the workshops, student confidence in discussing machine learning concepts with non-technical individuals improved, with the mean increasing to 2 (see Table 7.5). Every student felt at least slightly confident discussing machine learning concepts with non-technical individuals, and felt more confident in discussing machine learning with experts after the workshop. Most students who reported no confidence discussing with experts felt at least slightly confident after the workshop. During smaller discussions after the workshops, students expressed that just learning the basic building blocks of machine learning models helped demystify the "scary term", and interacting with accessible workshop tools gave them more confidence to discuss machine learning concepts. The workshops were successful at improving student confidence in discussing machine learning for individuals of different backgrounds.

Effectiveness of Tools

Students engaged with a number of public ML tools throughout the six workshops. These tools included Google's Quick Draw, Google's Teachable Machine, the PAC webtool, and the PAC extension for MIT App Inventor. The post-questionnaire assessed the usability of these public ML tools by asking students their level of enjoyment for each tool, and any fun points or points of confusion they had experienced. Overall, student sentiment around the public ML tools was quite positive, with each tool scoping above a mean of 2 on a 0-3 scale (see Table 7.6).

Many students wrote that the customizability of both PAC and the App Inventor application were fun to tweak around with, implying that students enjoyed exploring

Question	Mean	Range
How much do you agree with the following statement: I enjoyed using the Quick Draw web tool.	2.75	2-3
How much do you agree with the following statement: I enjoyed using the Teachable Machine web tool.	2.35	2-3
How much do you agree with the following statement: I enjoyed using the Personal Audio Classifier web tool.	2.35	2-3
How much do you agree with the following statement: I enjoyed using the Voice Authentication app.	2.1	1-3

Table 7.6: Student response breakdown for questions related to teaching tool effectiveness.

and building their own applications of audio classification. Some students struggled with the Voice Authentication app, particularly with exporting the app and achieving good performance on the mobile devices provided in class. Due to the limited functionality of student tablets, some models trained using the PAC webtool did not work as well on the tablets.

PAC Customizability and Improvements

To get further feedback on the PAC interface and how successful students were at exploring various applications of audio classification, we asked students for the most confusing aspect of the PAC webtool, the types of audio classification models explored, and any general feedback on the PAC interface. Many students did not note any confusing aspects. Three specific pain points include the inability to play back recorded voice clips when building models, the model export process, and the previously stated issues with running the Voice Authentication app.

General feedback was positive in nature. Students explicitly mentioned the ease of use of the interface and the aesthetic elements of the PAC UI. Some students provided suggestions for new feature additions, including the ability to play back audio after users record clips. Student responses showed that they were able to quickly grasp the PAC interface and build impactful audio classification models.

7.6 Discussion

Both the PAC webtool/PAC extension package and the PIC webtool/PIC extension package have been important additions to a suite of public ML tools for MIT App Inventor that bring machine learning to the forefront of K-12 computer science curricula. While both PIC and PAC have received positive feedback from the Boston Latin Academy workshops and the MIT App Inventor community, much work needs to be done to improve the usefulness of these tools and make these learning experiences more accessible. We plan to build on the curricula for the PIC and PAC workshops using feedback from the workshop surveys to make the lessons more technically involved. The Boston Latin Academy has also expressed interest in porting the workshops to a Mobile CSP approved curriculum that would be used in AP Computer Science classes across the country.

8

Future Work and Summary of Contributions

I will end this dissertation by discussing some interesting areas of future work emerging from my research experiences. These areas are: (1) general education and K-12 initiatives in ML; (2) what types of public ML tools should be created; and (3) some suggestions for improving technical ML education. I will also summarize the contributions I have made in this thesis.

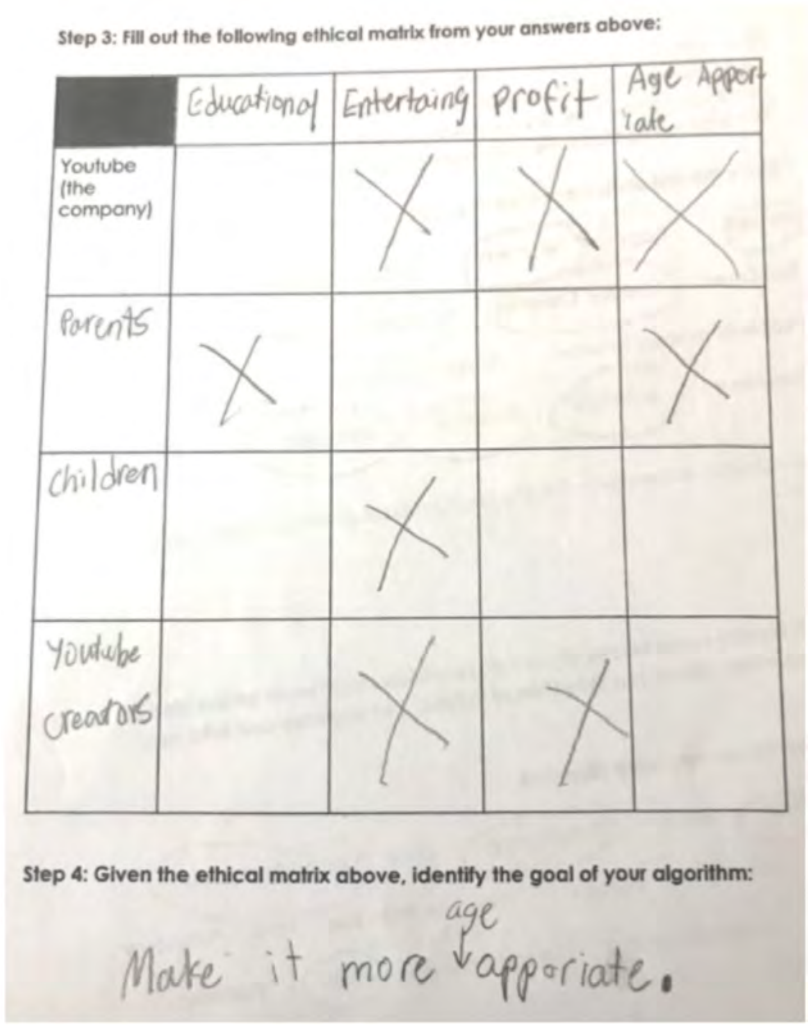
8.1 Creating more general education and K-12 programs

There are not enough easily accessible resources for general adult education or K-12 education in ML. The courses I discussed and analyzed in this thesis are great examples for these categories and we should help educators and curriculum developers make more of these courses. The new courses that we create should also relate more directly to societal needs: (1) these resources should help people understand the world they live in so that they are able to use ML products effectively and safely, and so they can identify and critically interpret ML outputs in their environment; (2) we should teach people to feel comfortable participating in ML advocacy in their local, national, and worldwide communities; and (3) these courses should teach people how to tinker with ML to create personal ML artifacts, which link well to technical courses and help more people potentially start upon career paths in ML.

From my analysis using the ML Education Framework (see Chapter 5), non-technical educational programs in general don't have enough focus on having students actually create ML artifacts. Scores for technical artifact creation (S3 from my framework) and societal/analytical artifact creation (S4 from my framework) were generally low for both K-12 and General Adult Courses in ML and AI. From education theory, these hands-on constructionist experiences are very important to developing understanding of ML and self-efficacy in ML, so future curricula should aim to include more such activities. Of course, the ability to include experiential learning activities with public ML tools (in particular, for technical ML artifact creation) requires classrooms to have access to computational devices that are able to run these tools. A variety of suitable tools for each audience type also needs to be made available to educators.

8.2 Designing better tools that allow more participation in creating technical and analytical ML artifacts

We need to design more public ML tools that allow for public participation in creating both technical and analytical ML artifacts. From my experiences, ML tools meant for general education need to be created in tandem with carefully designed curricula. Having technical tool creators collaborate with educators will help both parties consider how the tools will be used in a classroom setting and how to customize them for specific applications within learning experiences. It is also likely that different audiences will engage better with different learning experiences. For example, K-12 students and general adult audiences probably care about different types of ML applications. Public ML tools and curricula may need to be customized for the specific audience they are aimed towards.



Step 3: Fill out the following ethical matrix from your answers above:

	Educational	Entertaing	Profit	Age Appor-iate
Youtube (the company)		X	X	X
Parents	X			X
Children		X		
Youtube creators		X	X	

Step 4: Given the ethical matrix above, identify the goal of your algorithm:

Make it more age apporiate.

Figure 8-1: An ethical matrix for YouTube as analyzed by a middle school student [104].

There has been a lot of recent development of public ML tools for creating technical ML artifacts, but not enough work in building public ML tools for creating analytical ML artifacts. An excellent example of an analytical AI course is the AI + Ethics Curriculum for Middle School from the MIT Media Lab [104], which I analyzed using my framework (Section 5.2.6). In this course, students work with several paper-and-pencil tools such as ethical matrices to dive deeply into thinking about ML products as social, ethical, and technical systems (see Figure 8-1). These activities provide advanced coverage of S4. Analysis of ML Design Intentions and Results, from the ML Education Framework.

Tools such as ethical matrices and stakeholder analyses are great starting points for building more analytical tools for ML. Perhaps educators, ethicists, and technical ML developers can work to tailor such tools for specific ML applications and curricula. It will be interesting to explore if such analytical tools could be linked to or incorporated within technical tools to more directly support this type of analytical learning.

8.3 Improvements to technical ML education

From my analysis of technical courses using the ML Education Framework (see Chapter 5), they tend to cover relatively little regarding cultural, ethical, and societal considerations for building ML systems. The three items of K3. Bias in ML Systems, S4. Analysis of ML Design Intentions and Results, and S5. ML Advocacy were very lightly covered in technical curricula. The coverage of K4. Societal Implications of ML, also mostly focused on the many pros of machine learning without thoroughly discussing the difficulties of integrating new ML systems into existing human societies and processes or the potential differential harms that may be caused by ML systems. Most technical courses have one or two lectures on topics around fairness, bias, and responsible design and many courses don't cover these topics at all.

It is problematic that technical ML courses do not teach students to consider the whole ML pipeline (i.e. cultural, ethical, and societal perspectives). This needs to be amended so that the creators of ML technologies are more aware of the impact of their creations and can incorporate these considerations into the technical process of creating ML applications. From my experiences with Deep Learning Practicum, it is not enough to teach bias or fairness as one-off lectures. Students are unlikely to incorporate these into their own design processes unless they have previously practiced going through the pipeline in creating ML applications with such considerations in the past.

Ethicists and educators are starting to research how to modify the ML pipeline to make this happen [93]. We also need buy-in from tech companies, institutions, and the people who teach these technical courses. It is important to teach ML engineers and developers to become more holistic designers who are more aware of the potential impact of their creations.

8.4 Summary of Contributions

In this dissertation, I have presented an educational framework for transforming ML consumers to be ML contributors. The ML Education Framework contains 14 items in 3 categories: Knowledge (K1. General ML Knowledge, K2. Knowledge of ML Methods, K3. Bias in ML Systems, K4. Societal Implications of ML), Skills (S1. ML Problem Scoping, S2. ML Project Planning, S3. Creating ML Artifacts, S4. Analysis of ML Design Intentions & Results, S5. ML Advocacy, S6. Independent Out-of-Class Learning), and Attitudes (A1. Interest, A2. Identity & Community, A3. Self-Efficacy, A4. Persistence). This framework can be used in creating new ML curricula, in analyzing the results of ML interventions, and in situating support tools within ML education for students of a variety of backgrounds and levels. With respect to this framework:

- I designed and taught a semester long college course for ML tinkerers at Massachusetts Institute of Technology called 6.S198: Deep Learning Practicum. The course was project-based and taught in a laboratory setting through the Use-Modify-Create framework. Deep Learning Practicum focused on transfer learning techniques for deep learning, which helped students with little to no prior knowledge of programming or ML gain self-efficacy in creating personal ML artifacts. It transitioned students from using non-programming ML tools for hands-on exploration of ML applications to using TensorFlow.js to create ML applications.

- I created a Curriculum Rubric for analyzing how a course addresses each element of the ML Education Framework from a 0-4 scale. The Curriculum Rubric can be used by curriculum developers to determine which elements of the framework they wish to focus on at the beginning of the design process, by educators to evaluate how well their curriculum covers elements of the ML education framework, and by educators to choose which curricula and workshops they want to use for their specific goals. I analyzed a set of 6 technical, general, and K-12 ML/AI courses by others and provided insights into general trends in ML education.

- I created Personal Image Classifier (PIC) and Personal Audio Classifier (PAC), two public ML webtools for image and audio classification that can be directly used in creating mobile applications with MIT App Inventor. These tools use transfer learning in the browser to allow students as young as middle school train and test personalized image or audio classification models. These models can then be downloaded for use in MIT App Inventor, a blocks-based mobile application creation platform that allows people with no programming experience to quickly and easily create powerful mobile applications.

- I offered recommendations for ML education research in creating more general education and K-12 programs that target societal needs, in designing better public ML tools that allow for more participation in creating technical and analytical ML artifacts, and in including societal, ethical, and cultural considerations within the ML pipeline to improve technical ML education to train more holistic ML engineers and researchers.

Appendix A

Curriculum Rubric - ML Education Framework

A live version of the curriculum rubric for the ML Education Framework can be found at the reference linked in [69]. The below rubric has been updated as of August 28, 2020.

Category	4: Advanced coverage	3: Adequate coverage	2: Some coverage	1: Minimum coverage
***K1. General ML Knowledge**	Graduates of this course can give a precise definition of machine learning (e.g. "a model that can complete a specific task after being trained by humans to find patterns from large amounts of data") and provide a detailed description of the steps of the machine learning pipeline with technical and socio-ethical considerations for each step.	Graduates of this course can give an adequate definition of machine learning and provide a cursory overview of the machine learning pipeline.	Graduates of this course can give a superficial definition of machine learning (e.g. "teaching machines to do things that people can do") and provide a few accurate examples of machine learning applications (e.g. Google search, YouTube recommendations).	Graduates of this course recognize the term "machine learning" but demonstrate limited abilities in defining the term or providing examples of the technology (e.g. "machine learning has something to do with computers and technology").
K2. Knowledge of ML Methods	Graduates of this course are able to accurately discern when to use a range of machine learning methods across the breadth of the field. They are able to describe core technical concepts of these methods and comfortably use/implement them in appropriate applications. (For high school and above, a good range may include supervised methods such as K-nearest neighbors, CART/DT, regression, convolutional neural networks; unsupervised methods such as k-means clustering, principle component analysis, GANs, embeddings; RNNs/LSTMs; reinforcement learning; transfer learning; ensemble methods For middle school and below, the methods would likely be framed around high level applications that allow students to complete specific tasks using machine learning, including using GAN, RNN, or CNN applications to create art or music, using RL or hidden Markov models to play games...etc.)	Graduates of this course are able to accurately discern when to use several machine learning methods. They are able to describe core technical concepts of these methods and use/implement them in appropriate applications.	Graduates of this course are able to describe the use cases for and core technical concepts of one or two machine learning methods. They are also able to use/implement the method(s) in appropriate applications.	Graduates of this course are able to describe the use cases for and general workings of a few machine learning methods at a surface level.

***K3. Bias in ML Systems**	Graduates of this course are able to describe how machine learning systems may come to be unpredictably biased against specific groups throughout each step of the machine learning pipeline and critically incorporate the practices of ethical thinking and design in their own work.	Graduates of this course are able to describe how machine learning systems may come to be unpredictably biased against specific groups throughout each step of the machine learning pipeline.	Graduates of this course are able to detail one or two examples of bias in machine learning systems, describing the causes of bias and resulting harms (e.g. some examples of bias: Northpointe's COMPAS system used in criminal justice, hiring/resume screening models, fruit sorting models performing poorly on green apples vs red ones; some causes: incorrect assumptions, unbalanced training data, proxies, bad benchmarks/metrics, choice of loss function; some harms: hiring discrimination, disparate healthcare outcomes).	Graduates of this course are able to articulate that math-based models are not neutral and that it is possible for machine learning systems to create as biased outcomes as human decision-makers or even amplify bias.
***K4. Societal Implications of ML**	Graduates of this course recognize that it is necessary for creators of ML technologies to consider societal implications of their work and are able to apply ethical and cultural perspectives/concepts to analyses of machine learning artifacts in comprehensive, interrelational, and sensitive ways (i.e. considering privacy, security, potential for abuse, balance of benefits and harms, ethnographic reception, disparate impacts; using tools such as stakeholder analysis, ethical matrix, model cards...etc.).	Graduates of this course recognize that it is important for creators of ML technologies to consider societal implications of their work and are able to apply ethical and cultural perspectives/concepts to analyses of machine learning artifacts in not-fully-comprehensive ways.	Graduates of this course recognize that it is important for creators of ML technologies to consider societal implications of their work and are able to describe basic/obvious ethical or cultural impacts of ML technology (e.g. ML automating away jobs).	Graduates of this course do not recognize that it is important for ML engineers/creators to consider ethical, cultural, or societal consequences of their work, but are able to articulate that machine learning technologies exist within society and have societal implications.
***S1. ML Problem Scoping**	Graduates of this course are able to determine which problems can be solved using machine learning from a technical perspective and which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective.	Graduates of this course are able to determine which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective (e.g. unsuitable problems: predicting the trajectory of a rocket when it is more easily and accurately done using physics/non-stochastic methods; problems with high standards for explainability or interpretability) but are unable to determine which problems can be solved using machine learning from a technical perspective.	Graduates of this course are able to determine which problems can be solved using machine learning from a technical perspective (i.e. whether it is possible to obtain a lot of training data and if the problem is such that past data predicts future outcomes) but are unable to determine which problems should/n't be solved using machine learning from a logical, ethical, or cultural perspective.	Graduates of this course are able to recognize problems similar to exemplar ML problems they have seen in the past.

S2. ML Project Planning	For a range of ML problems, graduates of this course are able to plan out a solution sensitive to both technical implementation considerations (i.e. what kind of data is needed, how to obtain these datasets, what kind of methods/models to use, what computational resources are available for training, what metrics to use for analysis, how the solution fits into the delivery setting such as mobile vs. cloud, how to maintain a feedback loop into the system after deployment...etc.) and contextual factors including ethics and cultural dimensions (e.g. data privacy, security, how the solution fits into existing human, technical, and systemic processes, potential concerns of users or 'used upon').	For a range of ML problems, graduates of this course are able to plan out a solution with an adequate technical approach but minimal contextual considerations.	For one or two types of ML problems, graduates of this course are able to plan out a solution with adequate technical approach and contextual considerations.	For one or two types of ML problems, graduates of this course are able to plan out a solution with an adequate technical approach but minimal contextual considerations.
S3. Creating ML Artifacts	Graduates of this course are able to accurately identify appropriate machine learning tool(s) that can be used for a wide range of machine learning applications and comfortably use the tool(s) to implement the given application (e.g. Beautiful Soup and Kaggle for data collection; Teachable Machine 2, Cognimates, Keras, and PyTorch for model creation, training, and analysis; AWS and GCP for computation).	Graduates of this course are able to identify appropriate machine learning tool(s) that can be used for several types of machine learning applications and use the tool(s) to implement the given application.	Graduates of this course are able to identify appropriate machine learning tool(s) for one or two types of machine learning applications and use the tool(s) to implement the given application (e.g. using Teachable Machine to create new classification models or using the Embedding Projector to analyze different sets of data).	Graduates of this course are able to use machine learning tool(s) to replicate one or two machine learning applications from the course (e.g. using Fast Style Transfer webapp to create art).

***S4. Analysis of ML Design Intentions and Results**	Graduates of this course are able to extract both implicit and explicit design intentions of an ML system (i.e. based on affordances, use cases, accuracy targets of classes/sub-classes, socio-political intentions, financial incentives) and are able to critically analyze those initial intentions in relation to how the deployed system can and should be used (i.e. pros and cons of usage, when you should use something else, societal impacts...).	Graduates of this course are able to identify explicit design intentions of an ML system as well as extract a few implicit design intentions, and are able to analyze those initial intentions in relation to how the deployed system can be used.	Graduates of this course are able to identify explicit design intentions of an ML system and are able to relate those initial intentions to a few elements of the deployed system.	Graduates of this course are able to identify explicit design intentions of an ML system but are unable to relate these intentions to the behavior of the deployed system (e.g. able to identify that the Amazon website wants to sell items to make money, but unable to understand that a product recommender algorithm helps with that intention).
***S5. ML Advocacy**	Graduates of this course are able to critically engage with their communities regarding machine learning policies, products, and education. They feel motivated to do so and actively seek out opportunities (e.g. teaching family and friends about machine learning, voicing opinions about policies at a town hall).	Graduates of this course are able to critically engage with their communities regarding machine learning policies, products, and education. They feel that it is valuable to do so but do not actively seek out opportunities.	Graduates of this course are able to critically engage with others regarding machine learning policies, products, or education but do not feel it is important to do so.	Graduates of this course are able to engage with others at a surface level regarding machine learning policies, products, or education (e.g. they don't consider social processes or don't recognize ethical problems with ML).
S6. Independent Out-of-Class Learning	During this course, students actively seek out highly engaging learning experiences outside of the classroom (e.g. follow journals or blogs about advances in machine learning or experiment with new products and methods), and are able to independently learn new machine learning concepts and skills.	During this course, students actively seek out low-commitment learning experiences outside of the classroom (e.g. keeping up with news about the technology) and are able to independently learn new machine learning concepts or skills.	During this course, students do not seek out learning experiences outside of the classroom but are able to learn new machine learning concepts or skills not taught in the course (e.g. if their in-class project requires them to use a method not covered in the curriculum).	During this course, students do not seek out learning experiences outside of the classroom but are able to understand and draw connections to their prior knowledge when presented with information relating to machine learning (e.g. if they see a news article about a machine learning product).
A1. Interest	Graduates of this course are very interested in machine learning and feel highly motivated to engage more with the subject.	Graduates of this course are interested in machine learning and feel that they want to engage more with the subject.	Graduates of this course have some interest in machine learning and are unsure if they want to engage more with the subject.	Graduates of this course have a cursory interest in machine learning and do not plan to engage more with the subject.

A2. Identity and Community	Graduates of this course participate in a larger machine learning community/(ies) where they feel like they belong, can receive help from, and can give back to. They also feel like they have an active role in fostering and shaping the development of the machine learning community/(ies) that they are part of.	Graduates of this course participate in a larger machine learning community/(ies) where they feel like they belong, can receive help from, and can give back to.	Graduates of this course are aware of at least one larger machine learning tinkerer/learning community but do not participate (i.e. because they cannot access the community, do not feel like they belong, or do not wish to engage).	Graduates of this course participate in a small machine learning community of their peers during the course.
A3. Self-Efficacy	Graduates of this course feel fully empowered (highly motivated and prepared) to design and build new and meaningful machine learning artifacts.	Graduates of this course feel motivated to design and build new and meaningful machine learning artifacts, but do not feel entirely confident that they are able to succeed without dedicated resources.	Graduates of this course feel motivated to design and build new and meaningful machine learning artifacts, but feel that they would not be able to succeed without dedicated resources.	Graduates of this course do not feel motivated to design and build new and meaningful machine learning artifacts, but feel that they could potentially succeed if provided with dedicated resources (such as the ones from the course, including mentorship, technical support, compute, software...etc.).
A4. Persistence	Graduates of this course voluntarily and successfully decide to pursue or change career paths in order to work in a job relating to ML.	Graduates of this course voluntarily take more classes or do projects in computing, data science, or ML in the future (e.g. middle school students signing up for classes in high school, high school students deciding to major or minor in college) and succeed in these endeavors.	Graduates of this course voluntarily engage with machine learning at a surface level in the future (e.g. keep up with news about the technology, experiment with new products).	Graduates of this course engage with machine learning in the future out of necessity (i.e. for school or work) but would not if given the choice.
Instructional Design	The course was implemented with deliberate attention to good instructional design practices (i.e. well defined expected prior knowledge and course objectives, builds upon a clear learning progression throughout the course, properly links modules with few gaps, activities are engaging and have strong ties to learning outcomes, evaluations are well-defined and helpful to students, provides opportunities for feedback...etc.).	The course was implemented with adequate attention to good instructional design practices.	The course was implemented with some attention to good instructional design practices (e.g. thought was given to content sequencing).	The course was implemented with minimal awareness of good instructional design practices (e.g. material is unsuitable for the audience, unclear learning objectives, activities don't lead to intended learning outcomes).

Bibliography

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, page 265–283, USA, 2016. USENIX Association.

[2] Mahmoud Abdulwahed and Zoltan K. Nagy. Applying kolb's experiential learning cycle for laboratory education. *Journal of Engineering Education*, 98(3):283–294, 2009.

[3] Kimberly Adams. Smartphones play crucial role for black lives matter. `https://www.marketplace.org/2016/07/11/smartphones-play-crucial-role-black-lives-matter/`, Jul 2016.

[4] aik12 MIT. Innovating learning and education in the era of ai. `https://aieducation.mit.edu/`.

[5] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. May 2016.

[6] A. Bandura. *Encyclopedia of human behavior (Vol. 4)*, chapter Self-efficacy, pages 71–81. Academic Press, New York, 1994. Reprinted in H. Friedman [Ed.], Encyclopedia of mental health. San Diego: Academic Press, 1998.

[7] Albert Bandura. Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, 84(2):191–215, 1977.

[8] Albert Bandura, Claudio Barbaranelli, Gian Vittorio Caprara, and Concetta Pastorelli. Self-efficacy beliefs as shapers of children's aspirations and career trajectories. *Child Development*, 72(1):187–206, 2001.

[9] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks, 2018.

[10] Sergio Bermejo, Miguel García-González, Ramon Bragós, Nuria Montesinos, and Montserrat Ballarín. Steps for iterating design-implement experiences into a cdio course. Jun 2016.

[11] Nikhil Bhatia. Using transfer learning, spectrogram audio classification, and mit app inventor to facilitate machine learning understanding. Master's thesis, Massachusetts Institute of Technology, 2020.

[12] Nikhil Bhatia and Natalie Lao. Using transfer learning, spectrogram audio classification, and mit app inventor to facilitate machine learning understanding. In *The Fourth International Conference on Computational Thinking Education 2020*, CTE, 2020.

[13] Venkatesh Boddapati, Andrej Petef, Jim Rasmusson, and Lars Lundberg. Classifying environmental sounds using image recognition networks. *Procedia Computer Science*, 112:2048 – 2056, 2017. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

[14] Karen Brennan and Mitchel Resnick. New frameworks for studying and assessing the development of computational thinking. In *AERA*, 2012.

[15] AP Central. Ap computer science principles | ap central - the college board. `https://apcentral.collegeboard.org/courses/ap-computer-science-principles`, February 2019.

[16] Tara Chklovski, AnnaLise Hoopes, and Dara Olmsted. Technovation challenge: Teaching girls computer science entrepreneurship (abstract only). In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, page 747, New York, NY, USA, 2013. Association for Computing Machinery.

[17] European Commission. United kingdom ai strategy report. `https://ec.europa.eu/knowledge4policy/ai-watch/united-kingdom-ai-strategy-report_en`, Aug 2020.

[18] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 191–198, New York, NY, USA, 2016. Association for Computing Machinery.

[19] Edward F. Crawley. *The CDIO Syllabus: A Statement of Goals for Undergraduate Engineering Education.* Massachusetts Institute of Technology, 2001.

[20] Edward F. Crawley, Johan Malmqvist, William A. Lucas, and Doris R. Brodeur. The cdio syllabus v2.0: An updated statement of goals for engineering education. In *Proceedings of 7th international CDIO conference*, 2011.

[21] Matt Day, Giles Turner, and Natalia Drozdiak. Amazon workers are listening to what you tell alexa. *Bloomberg.com*, Apr 2019.

[22] deeplearn.js. font-explorer. `https://github.com/mintingle/font-explorer`, February 2018.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[24] Google Developers. Machine learning crash course. `https://developers.google.com/machine-learning/crash-course`.

[25] John Dewey. *Experience and Education.* Macmillan, 1938. Print.

[26] Graham Dove, Kim Halskov, Jodi Forlizzi, and John Zimmerman. Ux design innovation: Challenges for working with machine learning as a design material. May 2017.

[27] Ron Dror and Andrew Ng. Cs229: Machine learning. `http://cs229.stanford.edu/syllabus.html`, December 2018.

[28] S. Druga, Program in Media Arts, and Sciences (Massachusetts Institute of Technology). *Growing Up with AI: Cognimates : from Coding to Teaching Machines.* Massachusetts Institute of Technology, School of Architecture and Planning, Program in Media Arts and Sciences, 2018.

[29] Caitlin Duncan and Tim Bell. A pilot computer science and programming course for primary school students. In *Proceedings of the Workshop in Primary and Secondary Computing Education*, WiPSCE '15, page 39–48, New York, NY, USA, 2015. Association for Computing Machinery.

[30] Logan Engstrom. Fast style transfer. `https://github.com/lengstrom/fast-style-transfer/`, 2016.

[31] Technovation Families. Ai education for families. `https://www.curiositymachine.org/lessons/lesson/`.

[32] fast.ai. Practical deep learning for coders. `https://course.fast.ai/`.

[33] D. Fisher and N. Frey. *Better Learning Through Structured Teaching: A Framework for the Gradual Release of Responsibility.* Gale virtual reference library. Association for Supervision and Curriculum Development, 2008.

[34] Mozilla Foundation. Responsible computer science challenge. `https://foundation.mozilla.org/en/initiatives/responsible-cs/`.

[35] The K–12 Computer Science Framework. Use-modify-create trajectory. `https://k12cs.org/computational-thinking/`.

[36] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2018.

[37] Christina Gardner-McCune and David Touretzky. K-12 guidelines for artificial intelligence: What students should know. https://conference.iste.org/2019/program/search/detail_session.php?id=112142285, Jun 2019.

[38] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[39] Timnit Gebru. Oxford handbook on ai ethics book chapter on race and gender, 2019.

[40] GoOpenMichigan. Goopenmichigan. https://goopenmichigan.org/browse?batch_start=40&f.keyword=coding.

[41] Shuchi Grover and Roy Pea. Computational thinking in k–12: A review of the state of the field. *Educational Researcher*, 42:38–43, Feb 2013.

[42] David Ha and Douglas Eck. A neural representation of sketch drawings, 2017.

[43] David Ha, Jonas Jongejan, and Ian Johnson. Draw together with a neural network. https://magenta.tensorflow.org/assets/sketch_rnn_demo/index.html, January 2019.

[44] Adam Harley. 3d visualization of a convolutional neural network. http://scs.ryerson.ca/~aharley/vis/conv/, 2015.

[45] John Hartquist. fastai_audio. https://github.com/jhartquist/fastai_audio, Feb 2020.

[46] Harold Harty and Dwight Beall. Toward the development of a children's science curiosity measure. In *Journal of Research in Science Teaching*, volume 21 4, pages 425–36, 1984.

[47] Avi Hofstein and Vincent N. Lunetta. The laboratory in science education: Foundations for the twenty-first century. *Science Education*, 88(1):28–54, 2004.

[48] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[49] F. Hsu. *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton paperbacks. Princeton University Press, 2002.

[50] Ting-Chia Hsu. Ai 2 robot city [app and ai curriculum product]. https://play.google.com/store/apps/details?id=appinventor.ai_linlin777888999111.AI2_ROBOT_CITY_v2_0623, 2020.

[51] Karen Hussar, Sarah Schwartz, Ellen Boiselle, and Gil G. Noam. Toward a systematic evidence-base for science in out-of-school time.

[52] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[53] ReadyAI: Start AI in 10 minutes. Ai-in-a-box. `https://www.readyai.org/readyai-you/ai-in-a-box/`, May 2020.

[54] Kaggle Inc. Kaggle: Your home for data science. `https://www.kaggle.com/`, 2019.

[55] Google People + AI Research Initiative. deeplearn.js model builder demo. `https://courses.csail.mit.edu/6.s198/spring-2018/model-builder/src/model-builder/`, September 2018.

[56] MIT App Inventor. Explore mit app inventor. `https://appinventor.mit.edu/`.

[57] MIT App Inventor. Introduction to machine learning: Image classification. `http://appinventor.mit.edu/explore/resources/ai/image-classification-look-extension`.

[58] MIT App Inventor. Mole mash. `https://appinventor.mit.edu/explore/ai2/molemash`.

[59] MIT App Inventor. Personal audio classifier. `https://appinventor.mit.edu/explore/resources/ai/personal-audio-classifier`.

[60] MIT App Inventor. Personal image classifier. `https://appinventor.mit.edu/explore/resources/ai/personal-image-classifier`.

[61] MIT App Inventor. App inventor extensions. `http://ai2.appinventor.mit.edu/reference/other/extensions.html`, Jun 2017.

[62] Aylin Caliskan Islam, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora necessarily contain human biases. *CoRR*, abs/1608.07187, 2016.

[63] Ken Kahn and Niall Winters. Child-friendly programming interfaces to ai cloud services. pages 566–570, Sep 2017.

[64] Andrej Karpathy. Recurrentjs sentence memorization demo. `https://cs.stanford.edu/people/karpathy/recurrentjs/`, May 2015.

[65] A. Khamparia, D. Gupta, N. Nguyen, A. Khanna, B. Pandey, and Prayag Tiwari. Sound classification using convolutional neural network and tensor deep stacking network. *IEEE Access*, 7:7717–7727, 2019.

[66] David A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall, Inc., Englewood Cliffs, N.J, 1984. Print.

[67] David A. Kolb. *Experiential Learning: Experience as the Source of Learning and Development*. Pearson FT Press, New Jersey, 2014. Print.

[68] Dale Lane. Machine learning for kids. `https://machinelearningforkids.co.uk/`.

[69] Natalie Lao. Curriculum rubric - ml education framework. `https://docs.google.com/spreadsheets/d/15eLf50JBkD8ZGiBEULplP4c5uyRQA3tkUMwbz1vcWlg/edit#gid=0`.

[70] Natalie Lao and Hal Abelson. Deep learning practicum. `http://mit.edu/6.s198`.

[71] Natalie Lao, Irene Lee, and Hal Abelson. A deep learning practicum: Concepts and practices for teaching actionable machine learning. In *12th Annual International Conference of Education, Research and Innovation*, ICERI'19. International Academy of Technology, Education and Development (IATED), 2019.

[72] Natalie Lao, Irene Lee, and Hal Abelson. Experiences from teaching actionable machine learning at the university level through a small practicum approach. In *The Fourth International Conference on Computational Thinking Education 2020*, CTE, 2020.

[73] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. Computational thinking for youth in practice. *ACM Inroads*, 2:32–37, Feb 2011.

[74] Kurt Lewin. *Field Theory in Social Science: Selected Theoretical Papers*. Harpers, Oxford, England, 1951. Print.

[75] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *CoRR*, abs/1411.7766, 2014.

[76] D.C. Llach. Builders of the vision: Software and the imagination of design. pages 1–199, 2015.

[77] Aleksandra Luszczynska and Ralf Karl Schwarzer. *Predicting and changing health behaviour: Research and practice with social cognition models (3rd edition)*, chapter Social cognitive theory, pages 225–251. McGraw Hill, United States of America, 2015.

[78] Fred Martin, Irene Lee, Nicholas Lytle, Sue Sentance, and Natalie Lao. Extending and evaluating the use-modify-create progression for engaging youth in computational thinking. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 807–808, New York, NY, USA, 2020. Association for Computing Machinery.

[79] Santosh Maurya and Olivier AMMOUN. Implementing cdio approach in integrated digital environment. Jun 2018.

[80] MGHPCC. The massachusetts green high performance computing center (mghpcc). https://www.mghpcc.org/, 2018.

[81] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[82] M. Minsky and J. Lee. *Society Of Mind*. Touchstone book. Simon & Schuster, 1988.

[83] Mehryar Mohri. Foundations of machine learning – csci-ga.2566-001. https://cs.nyu.edu/~mohri/ml18/, December 2018.

[84] Ipsos MORI. Public views of machine learning: Findings from public research and engagement conducted on behalf of the royal society. Apr 2017.

[85] Paul Mozur. Beijing wants a.i. to be made in china by 2030. *The New York Times*, Jul 2017.

[86] Vineeth G. Nair. *Getting Started with Beautiful Soup*. Packt Publishing, 2014.

[87] Reiichiro Nakano. Gan playground - explore generative adversarial nets in your browser. https://reiinakano.github.io/gan-playground/, November 2017.

[88] Reiichiro Nakano. deeplearn.js style transfer demo. https://reiinakano.github.io/fast-style-transfer-deeplearnjs/, November 2018.

[89] Andrew Ng. Machine learning by stanford university. https://www.coursera.org/learn/machine-learning.

[90] Andrew Ng and deeplearning.ai. Ai for everyone. https://www.coursera.org/learn/ai-for-everyone.

[91] Thanh Thi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection: A survey. 2019.

[92] The University of Helsinki and Reaktor. Elements of ai. https://www.elementsofai.com/.

[93] Department of Linguistics and Massachusetts Institute of Technology Philosophy. Ethics ai. http://philosophy.mit.edu/ethicsandai/, Apr 2020.

[94] Massachusetts Institute of Technology. The mit quest for intelligence. `https://quest.mit.edu/about/`, 2018.

[95] Executive Office of the President of the United States. *Summary of the 2018 White House Summit on Artificial Intelligence for American Industry.* The White House, May 2018.

[96] The Partnership on AI. Report on algorithmic risk assessment tools in the u.s. criminal justice system. Apr 2019.

[97] Spectroscopy Online. William henry fox talbot and the foundations of spectrochemical analysis. Mar 2013.

[98] OpenAI. Gym. `https://gym.openai.com/`, 2019.

[99] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[100] Seymour Papert. *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books, Inc., USA, 1980.

[101] Adam Paszke, S. Gross, Soumith Chintala, G. Chanan, E. Yang, Zachary Devito, Zeming Lin, Alban Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[102] Evan W. Patton, Michael Tissenbaum, and Farzeen Harunani. *MIT App Inventor: Objectives, Design, and Development*, pages 31–49. Springer Singapore, Singapore, 2019.

[103] Blakeley H. Payne. Mit ai ethics education curriculum. `https://docs.google.com/document/d/1e9wx9oBg7CR0s5O7YnYHVmX7H7pnITfoDxNdrSGkp60/edit#heading=h.ictx1ljsx0z4`, Aug 2019.

[104] Blakeley H. Payne. Can my algorithm be my opinion?: An ai + ethics curriculum for middle school students. Master's thesis, Massachusetts Institute of Technology, 2020.

[105] Christine McLeavey Payne. Musenet. `https://openai.com/blog/musenet/`, Apr 2019.

[106] Jean Piaget and Barbel Inhelder. *The Psychology of the Child.* Basic Books, New York, 1969. Print.

[107] Juan Rodríguez, Jesús Moreno-León, Marcos Román-González, and Gregorio Robles. Learningml: A tool to foster computational thinking skills through practical artificial intelligence projects. *Revista de Educación a Distancia (RED)*, 20, Apr 2020.

[108] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[109] D. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[110] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, USA, 3rd edition, 2009.

[111] Arthur L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress*, pages 366–400. Springer New York, New York, NY, 1988.

[112] Flora Sapio, Weiming Chen, and Adrian Lo. China's new generation of artificial intelligence development plan. *FLIA: Foundation for Law International Affairs*, Jul 2017.

[113] National Science and Media Museum. The history of photography in pictures. `https://www.scienceandmediamuseum.org.uk/objects-and-stories/history-photography`.

[114] R. Benjamin Shapiro, Rebecca Fiebrink, and Peter Norvig. How machine learning impacts the undergraduate computing curriculum. *Commun. ACM*, 61(11):27–29, October 2018.

[115] Rohit Singh, Tommi Jaakkola, and Ali Mohammad. 6.867 machine learning. `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/#`, 2006.

[116] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, Stan Bileschi, Michael Terry, Charles Nicholson, Sandeep N. Gupta, Sarah Sirajuddin, D. Sculley, Rajat Monga, Greg Corrado, Fernanda B. Viégas, and Martin Wattenberg. Tensorflow.js: Machine learning for the web and beyond, 2019.

[117] Jackie Snow. "we're in a diversity crisis": cofounder of black in ai on what's poisoning algorithms in our lives. *MIT Technology Review*, Feb 2018.

[118] Stoj.io, Use All Five, Creative Lab, and PAIR teams at Google. Teachable machine. `https://teachablemachine.withgoogle.com/`, September 2018.

[119] Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R. Cooperstock. Can you teach me to machine learn? In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, SIGCSE '19, pages 948–954, New York, NY, USA, 2019. ACM.

[120] Danny Tang. Empowering novices to understand and use machine learning with personalized image classification models, intuitive analysis tools, and mit app inventor. Master's thesis, Massachusetts Institute of Technology, 2019.

[121] Danny Tang, Yuria Utsumi, and Natalie Lao. Pic: A personal image classification webtool for high school students. In *International Workshop on Education in Artificial Intelligence K-12*, EduAI '19, 2019.

[122] Jari Tanner. Finland offers crash course in artificial intelligence to eu. `https://federalnewsnetwork.com/world-news/2019/12/finland-offers-crash-course-in-artificial-intelligence-to-eu/`, Dec 2019.

[123] Technovation. About | technovation families. `https://www.curiositymachine.org/about/`.

[124] Technovation. Meet the first ai world championship winners! `https://www.technovation.org/blogs/meet-the-first-ai-world-championship-winners/`.

[125] TensorFlow. Embedding projector - visualization of high-dimensional data. `http://projector.tensorflow.org/`, December 2018.

[126] TensorFlow.js. Webcam pacman. `https://storage.googleapis.com/tfjs-examples/webcam-transfer-learning/dist/index.html`.

[127] thibo73800. Metacar: A reinforcement learning environment for self-driving cars in the browser. `https://www.metacar-project.com/`, 2019.

[128] Vincent Tinto. From retention to persistence. `https://www.insidehighered.com/views/2016/09/26/how-improve-student-persistence-and-completion-essay`, Sep 2016.

[129] Mike Tissenbaum, Josh Sheldon, and Hal Abelson. From computational thinking to computational action. *Communications of the ACM*, 62:34–36, Feb 2019.

[130] Dave Touretzky. ai4k12. `AI4K12.org`.

[131] David Touretzky. Developing national guidelines for teaching ai in k-12. `http://eduai19.ist.tugraz.at/index.php/workshop-program`, Jul 2020.

[132] Harvard University. Embedded ethics @ harvard. `https://embeddedethics.seas.harvard.edu/`.

[133] M. Varelas. *Identity construction and science education research: Learning, teaching, and being in multiple contexts (Vol. 35)*. Springer Science Business Media, 2012. Print.

[134] L.S. Vygotsky. *Mind in Society*. Harvard University Press, Cambridge, MA, 1978. Print.

[135] Molly H. Weinburgh and Donald Steele. The modified attitudes toward science inventory: Developing an instrument to be used with fifth grade urban students. In *Journal of Women and Minorities in Science and Engineering*, volume 6 1, pages 87–94, 2000.

[136] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viegas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE Transactions on Visualization and Computer Graphics*, page 1–1, 2019.

[137] Jeannette Wing. Computational thinking and thinking about computing. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 366:3717–25, Nov 2008.

[138] Hsin-Kai Wu and Ya-Ling Huang. Ninth-grade student engagement in teacher-centered and student-centered technology-enhanced learning environments. *Science Education*, 91(5):727–749, 2007.

[139] Raymond A. Yeh, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with perceptual and contextual losses. *CoRR*, abs/1607.07539, 2016.

[140] Catherine Yeo. Best k-12 resources to teach ai ethics. May 2020.

[141] Franziska Zimmer, Katrin Scheibe, and Wolfgang Stock. Echo chambers and filter bubbles of fake news in social media. man-made or produced by algorithms? In *8th Annual Arts, Humanities, Social Sciences Education Conference*, Dec 2018.