HDSR
HARVARD DATA SCIENCE REVIEW

Volume 0 Issue 0
DOI: 00.000
ISSN: 2644-2353

Widening Access to Applied Machine Learning with TinyML

Vijay Janapa Reddi[†,*] Brian Plancher[†,*] Susan Kennedy[†] Laurence Moroney[‡] Pete Warden[‡] Lara Suzuki[‡] Anant Agarwal[◇,§] Colby Banbury[†] Massimo Banzi[‖] Matthew Bennett[◇] Benjamin Brown[†] Sharad Chitlangia[†] Radhika Ghosal[†] Sarah Grafman[†] Rupert Jaeger[⊥] Srivatsan Krishnan[†] Maximilian Lam[†] Daniel Leiker[⊥] Cara Mann[◇] Mark Mazumder[†] Dominic Pajak[‖] Dhilan Ramaprasad[†] J. Evan Smith[†] Matthew Stewart[†] Dustin Tingley[†]

[†] Harvard University [‡] Google [◇] edX [§] MIT [‖] Arduino [⊥] CreativeClass.ai

Abstract. Broadening access to both computational and educational resources is critical to diffusing machine-learning (ML) innovation. However, today, most ML resources and experts are siloed in a few countries and organizations. In this paper, we describe our pedagogical approach to increasing access to applied ML through a massive open online course (MOOC) on Tiny Machine Learning (TinyML). We suggest that TinyML, applied ML on resource-constrained embedded devices, is an attractive means to widen access because TinyML both leverages low-cost and globally accessible hardware, and encourages the development of complete, self-contained applications, from data collection to deployment. To this end, a collaboration between academia and industry produced a four-part MOOC that provides application-oriented instruction on how to develop solutions using TinyML. The series is openly available on the edX MOOC platform, has no prerequisites beyond basic programming, and is designed for global learners from a variety of backgrounds. It introduces pupils to real-world applications, ML algorithms, data-set engineering, and the ethical considerations of these technologies through hands-on programming and deployment of TinyML applications in both the cloud and on their own microcontrollers. To facilitate continued learning, community building, and collaboration beyond the courses, we launched a standalone website, a forum, a chat, and an optional course-project competition. We also open-sourced the course materials, hoping they will inspire the next generation of ML practitioners and educators and further broaden access to cutting-edge ML technologies.

Keywords: Applied ML, TinyML, STEM Education, Edge AI, MOOC

[*]{vj@eecs, brian_plancher@g}.harvard.edu

Media Summary

Removing barriers and widening access to machine learning educational resources is vital for empowering future generations as the world adapts to the widespread and rapid adoption of AI. Such resources and expertise are currently concentrated in a handful of universities and corporations, resulting in a lack of accessibility for the broader public. To provide a more global outreach, a four-part freely available online course series was developed through a collaboration between academia and industry on the topic of tiny machine learning (TinyML). This field of machine learning is naturally oriented towards accessibility due to requiring only low-cost and globally accessible hardware, with a minimal technical background in computer science or machine learning necessary. The course allows students to focus on developing data-sets, algorithms, and ethical considerations by learning through real-world applications. Furthermore, a set of resources, including a forum, chat, project competition, and website, were set up to facilitate continued learning, networking, and collaboration among former participants. We hope that these resources inspire additional efforts to make machine learning accessible to all, regardless of geographic or socioeconomic background.

1. Introduction

The past two decades have seen machine learning (ML) progress dramatically from a purely academic discipline to a widespread commercial technology that serves a range of sectors. ML allows developers to improve business processes and human productivity through data-driven automation. Given the ubiquity and success of ML applications, its commercial use is expected only to increase. Existing ML applications cover a wide spectrum that includes digital assistants (Maedche et al., 2019; T. M. Mitchell et al., 1994), autonomous vehicles (C. Chen et al., 2015; Dogan et al., 2011), robotics (A. Zeng et al., 2019), health care (Ahmad et al., 2018), transportation (Jahangiri & Rakha, 2015; Zantalis et al., 2019), security (Buczak & Guven, 2015), and education (Alenezi & Faisal, 2020; Kolachalama & Garg, 2018), with a continual emergence of new use cases.

The proliferation of this technology and associated jobs have great potential to improve society and uncover new opportunities for technological innovation, societal prosperity, and individual growth. However, integral to this vision is the assumption that everyone, globally, has unfettered access to ML technologies, which is not necessarily the case. Expanding access to applied ML faces three significant challenges that must be overcome. First is a shortage of ML educators at all levels (Brown, 2019; Gagné, 2019). Second is insufficient resources, as training and running ML models often requires costly, high-performance hardware, especially as data sets continue to grow in size. Third is a growing gap between industry and academia, as even the best academic institutions and research labs are struggling to keep pace with the rapid progress and needs of industry.

Addressing these critical issues requires innovative education and workforce training programs to prepare the next generation of applied-ML engineers. To that end, this paper presents a pedagogical approach that cross-cuts from research to education to address these challenges and thereby increase global access to applied ML. Developed as an academic and industry collaboration, the resulting massive open online course (MOOC), TinyML on edX, not only teaches hands-on applied ML through the lens of real-world Tiny Machine Learning (TinyML) applications, but also considers the ethical and life-cycle challenges of industrial product development and deployment (see Figure 1). This effort is built on five guiding principles:

(1) Focus on **application-based pedagogy** that covers all ML aspects; Instead of isolated theory and ML-model training, show how to physically design, develop, deploy, and manage trained ML models.

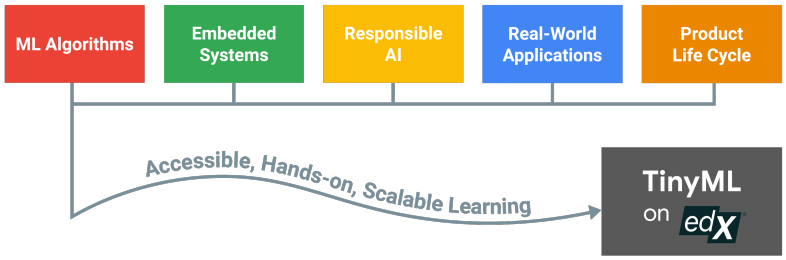


Figure 1. We designed a new applied-ML course motivated by real-world applications, covering not only the software (algorithms) and hardware (embedded systems) but also the product life cycle and responsible AI considerations needed to deploy these applications. To make it globally accessible and scalable, we focused on the emerging TinyML domain and released the course as a MOOC on edX.

(2) Through academic and industry collaboration, aid ML learners in **developing the full-stack, end-to-end skills** needed today, tomorrow, and in the foreseeable future.

(3) **Raise awareness of the ethical challenges** associated with ML aand familiarize learners with ethical reasoning skills to identify and address these challenges.

(4) **Prioritize accessibility** to students worldwide by teaching ML at a global scale through an open-access MOOC platform, using low-cost hardware that is available globally.

(5) **Build community** by providing a variety of platforms that allow participants to learn collaboratively and showcase their work.

We focus on teaching applied ML through the lens of TinyML as it enables easier global access to hands-on, end-to-end, application-based instruction. We define TinyML as a fast-growing field of machine learning technologies and applications that includes algorithms, hardware, and software capable of performing on-device sensor data analytics at extremely low power consumption, typically in the mW range and below, and hence enabling a variety of always-on ML use-cases on battery-operated devices. TinyML sits at the intersection of embedded systems and machine learning, and is targeted at enabling new embedded device applications (Figure 2). TinyML focuses on deploying simple yet powerful models onto microcontrollers at the network edge. These microcontrollers consume very low amounts of power ($<$ 1mW, 1000x lower than smartphones), and have very low cost (US$0.50 when ordered in bulk, orders of magnitude cheaper than conventional microprocessors such as those found in smartphones, desktops and servers). Since TinyML can run on microcontroller development boards with extensive hardware abstraction, the barrier to integrating an application with hardware is minimal. TinyML, therefore, enables a myriad of applications from agriculture, to conservation, to industrial predictive-maintenance. TinyML thus provides an attractive and strong entry point into applied ML. Because TinyML systems are powerful enough for many commercial tasks, learners can acquire full-stack developer skills that apply to their future jobs in relevant adjacent areas. For example, the lessons learned from end-to-end TinyML application design, development, deployment, and management are transferable to working with large-scale ML systems and applications, such as those found in data centers and portable mobile devices, smartphones, and laptops. Importantly, TinyML requires little data, and model training can employ simple procedures. As such, unlike with large-scale ML, it is possible for learners to complete the full applied ML workflow, from data collection to deployment.

Understanding ethical reasoning is a crucial skill for ML engineers as inaccurate or unpredictable model performance can erode consumer trust and reduce the chance of success. To this end, we

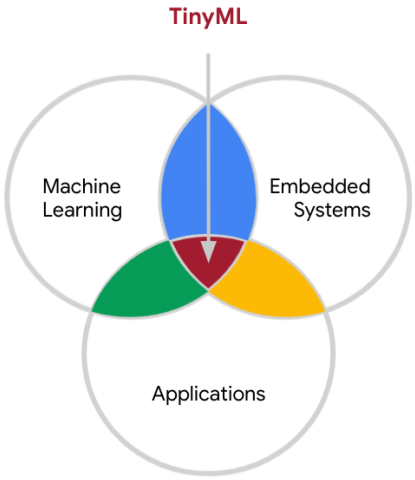


Figure 2. Tiny Machine Learning (TinyML) is focused on enabling new applications through the deployment of machine learning models onto embedded systems.

collaborated with the Harvard Embedded EthiCS program to integrate a responsible-AI curriculum into each course, providing opportunities to practice identifying ethical challenges and thinking through potential solutions to concrete problems based on real-world case studies.

Widening access requires both a globally accessible instructional platform and ubiquitous hardware resources that let users benefit from instructional resources at a minimal cost. We, therefore, deployed our pedagogical approach on edX, a MOOC provider that hosts university-level courses across many disciplines. Instruction is based in the free-to-use Google Collaboratory cloud programming environment and on the widely available low-cost microcontroller, the Arm Cortex-M microcontroller. We also worked with Arduino, an open-source hardware and software company, that designs and manufactures microcontrollers and microcontroller kits for building digital devices, to develop a low-cost, globally accessible, all-in-one TinyML Kit.

To foster collaboration and continued learning beyond this edX course, we developed a standalone website, discourse forum, discord chat, and an optional course-project competition. We hope the approach we devised brings ML to more people, not only to learn but also to teach applied ML and data science engineering principles to a broader audience. To that end, we have made all of our resources publicly available at `https://github.com/tinyMLx/courseware` and launched a broader TinyMLedu open education initiative that supports numerous outreach activities (see Section 8).

We launched the core TinyML edX series, comprising three sequential courses, between October 2020 and March 2021; an optional fourth course is under development. On average, more than 1,000 new students enroll each week. After a year, over 60,000 have enrolled from 176 countries. They come from diverse backgrounds and experiences, ranging from complete novices to experts who want to master an emerging field. Feedback suggests this strong enrollment may owe to the unique collaborative structure we foster between students, teachers, and industry leaders. Shared ownership between academia and industry appears to give participants confidence they are gaining skills that industry needs both today and tomorrow. Moreover, we recognize that opportunities to interact with experts is both encouraging and validating.

As part of our effort to widen access to applied ML, we have designed this paper to target multiple audiences. As such, different sections of the paper will be more or less relevant to different target audiences. Sections 2 and 3 are broadly applicable as they help to outline the opportunities and challenges associated with TinyML as a sub-discipline of ML, as well as its instruction. Sections 4

and 5, detailing the course structure and content, are largely targeted at both prospective students and teaching practitioners to help outline the pedagogical approach taken. Section 6 focuses on individuals involved in multimedia development, as well as accessibility researchers, helping outline how the course was developed with remote learning and accessibility in mind. The remainder of the sections, which outline future initiatives, demographic information associated with course participants, limitations, and related work, are broadly applicable to all of the above audiences.

2. Challenges and Opportunities of Applied-ML

We propose three criteria to empower applied-ML practitioners. First, we believe that no one size fits all with regard to interest, experience, and motivation, especially when broadening participation. Second, given ML innovation's breakneck pace, academic/industrial collaboration on cutting-edge technologies is paramount. Third, learners who wish to prepare for ML careers need experience with the entire development process from data collection to deployment. Moreover, they must understand the ethical implications of their designs before deploying them into the wild.

2.1. **Student Background Diversity.**

A major challenge in expanding ML access is that participants begin applied-ML courses with diverse background knowledge, work experience, learning styles, and goals. Hence, we must provide multiple on-ramps to meet their needs.

Participants include high-school and university students who want to learn about AI to develop cutting-edge applications and gain an edge in their careers, as many employers are increasingly expecting new hires to have some ML background be it either theoretical or applied in practice. Other participants are industry veterans looking to either pivot their careers toward ML or study the landscape of the TinyML field. For example, some are computer-systems engineers who want to learn about ML in general. Others are ML engineers and data scientists who want to expand their applied skills. Others are doctors, scientists, or environmentalists who are curious about how TinyML technology could transform their fields. Other participants are self-taught, makers, tinkerers, and hobbyists who want to build smart "things" based on emerging technologies.

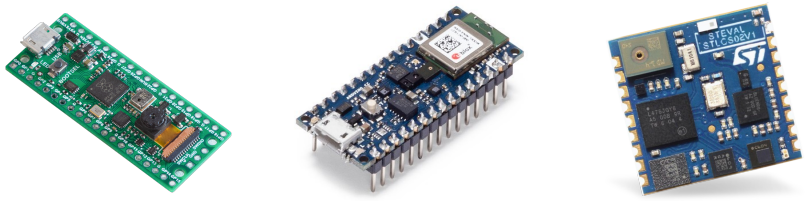
Given this broad spectrum, we have a unique opportunity to enable inclusive learning for all despite differing backgrounds and expertise. But we must provide multiple on-ramps. Specifically, we chose to structure the course in a spiral that sequentially addresses the same concepts with increasing complexity (Harden, 1999). Doing so ensures that not only do participants reinforce fundamentals while picking up new details, but they also master important objectives at every stage. This approach has been shown to improve learning while meeting each individual's objectives (Neumann et al., 2017).

2.2. **Need for Academia/Industry Collaboration.**

Expanding ML access requires the expertise of academia and industry. Academia is strong in structured teaching: it creates in-depth, rigorous curricula to impart a deep understanding of a field. Conversely, industry is more pragmatic, developing the skills necessary for employment. These approaches are complementary.

Also, ML is moving rapidly thanks largely to industry's access to rich data. Analysis of ML-research publications at the NeurIPS ML conference suggests industry leads ML innovation (Chuvpilo, 2020). As such, industry has essential domain-specific knowledge that helps ground ML pedagogy in practical skills and real-world applications.

We belive that academia and industry must work in tandem to deliver high-quality, accessible, foundational, and skills-based ML content. Joining a strong academic institution and a industry leader in technology innovation, with a history of releasing free and accessible resources, makes students confident that they are learning relevant skills for their careers.



(A) (B) (C)

Figure 3. Example TinyML devices: (a) Pico4ML, (b) Arduino Nano 33 BLE Sense, and (c) STMicroelectronics Sensor Tile. These and many other embedded devices are being built with integrated sensors, specifically for enabling on-device real-time data processing.

2.3. **Demand for Full-Stack ML Expertise.** In ML, the "full stack"[1] approach to building and using ML models is the core skill that will define future engineers. The engineers who bring long-term value to their industry are those who have the in-depth knowledge to innovate beyond well-known applications and scenarios. In fact, full-stack developers are now more numerous than all other developers combined, with 55% of developers identifying as full-stack in a 2020 report ("Stack Overflow Developer Survey", 2020).

Our academia and industry collaboration can ensure the course series imparts the full-stack abilities that industry demands. Doing so requires content beyond the narrow, well-lit path of ML-model training, optimization, and inference. We therefore also focus on acquiring and cleansing data, deploying models in hardware, and managing continuous model updates on the basis of field results. Our hope is that learners will gain a whole new set of applied-ML skills that they can leverage in their varied future careers.

3. ML's Future Is Tiny and Bright

We employ Tiny Machine Learning (TinyML), a cutting-edge applied-ML field that brings the potential of ML to low-cost, low-performance, and power-constrained embedded systems and thereby enables hands-on learning. TinyML lets us impart ML-application design, development, deployment, and life-cycle-management skills.

3.1. **Introduction to TinyML.** TinyML refers to the deployment of ML resources on small, resource-constrained devices (Figure 3). It starkly contrasts with traditional ML, which increasingly focuses on large-scale implementations that are often confined to the cloud. TinyML is neither a specific technology nor a method per se, but it acts in many ways as a proto-engineering discipline that combines machine learning, embedded systems, and performance engineering. Similar to how chemical engineering evolved from chemistry and how electrical engineering evolved from electromagnetism, TinyML has evolved from machine learning in cloud and mobile computing systems.

The TinyML approach dispels the barriers of traditional ML, such as the high cost of suitable computing hardware and the availability of data. As Table 1 shows, TinyML systems are nearly two to three orders of magnitude cheaper and more power efficient than traditional ML systems. As such, this approach can reduce the cost of ML and can handle tasks that go beyond traditional ML. The TinyML approach also makes it easy to emphasize responsible AI (Section 5).

[1]The term *full-stack* comes from historic career growth in web technologies and the Internet. It began as a series of loosely linked skills but now encompasses web development from the lowest level, the server, to the highest level, the web browser or mobile app.

Table 1. Cloud & Mobile ML systems versus TinyML systems.

Platform	Architecture	Memory	Storage	Power	Price
Cloud E.g., Nvidia V100	GPU Nvidia Volta	HBM 16GB	SSD/disk TB–PB	250W	∼$9,000
Mobile E.g., cellphone	CPU Arm Cortex-A78	DRAM 4GB	Flash 64GB	∼8W	∼$750
Tiny E.g., Arduino Nano 33 BLE Sense	MCU Arm Cortex-M4	SRAM 256KB	eFlash 1MB	0.05W	$3

TinyML supports large-scale, distributed, and local ML tasks. Inference on low-cost embedded devices allows scalability, and their low power consumption enables their operation in remote locations far from the electric grid. Since the number of tiny devices in the wild far exceeds the number of traditional cloud and mobile systems (IC Insights, 2020), TinyML is a prime candidate for local ML tasks that were once prohibitively expensive, such as distributed sensor networks and predictive maintenance systems in industrial manufacturing settings.

TinyML applications are broad and continue to expand as the field gains traction. The approach's unique value stems primarily from bringing ML close to the sensor, right where the data stream originates. Therefore, TinyML permits a wide range of new applications that traditional ML cannot deliver because of bandwidth, latency, economics, reliability, and privacy (BLERP) limitations.

Common TinyML applications include keyword spotting, visual wake words, and anomaly detection. Keyword spotting generally refers to identification of words that typically act as part of a cascade architecture to kick-start or control a system, such as a mobile phone responding to voice commands (Choi et al., 2019; Y. Zhang et al., 2017). Visual wake words involve parsing image data to find an individual (human or animal) or object. This task can potentially serve in security systems (Buczak & Guven, 2015), intelligent lighting (Gopalakrishna et al., 2012), wildlife conservation (Di Minin et al., 2018; Duhart et al., 2019), and more. Anomaly detection looks for abnormalities in persistent activities (Chandola et al., 2009). It has many applications in both consumer and commercial markets, such as checking for abnormal vibrations (Turnbull et al., 2021) or temperatures (X. Zeng et al., 2020) to provide early warnings of potential failures and to enable preventive maintenance (Gupta et al., 2015; Yairi et al., 2006).

3.2. **TinyML for Applied ML.** An applied-ML engineer should have full-stack experience to appreciate the end user impact of the various stages of the ML-development workflow. In prototypical ML, such as training large neural-network models in the cloud, learners are unable to participate locally in end-to-end ML development. For example, it is impossible to require them to collect millions of images (akin to ImageNet (Deng et al., 2009)) for large and complex tasks, such as general image classification. Even more difficult is asking all learners to buy the computational resources to train a complex ML model and then evaluate its performance in the real world.[2]

By contrast, the small form factor and domain-specific tasks of TinyML enable the full ML workflow, starting from data collection and ending with model deployment on embedded devices. Students thereby gain a unique experience. For example, to implement keyword spotting in their native language, course participants learn to collect their own speech data, train a model on that data, deploy it in an embedded device, and test the device in their community.

[2]A model may perform well on a test data set and still perform poorly in the real world.

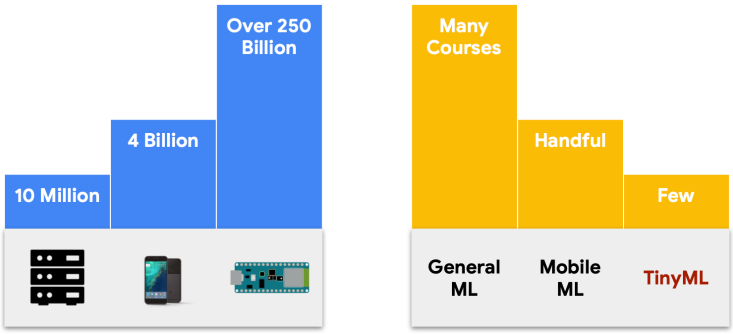


Figure 4. The number of available ML courses is disproportionate to the number of systems.

Such activities create an immersive learning experience, and they are feasible with TinyML because they only require about 30–40 samples[3] of spoken keywords which is easy to collect (only from people with their explicit consent) using a laptop with a web browser and microphone. Learners can then train the model using Google's free Colab environment (Bisong, 2019) and deploy it in a TinyML device using TensorFlow Lite for Microcontrollers (David et al., 2020) or another open-source software technology. This approach allows small keyword-spotting models (about 16KB) to run efficiently on low-cost, highly constrained hardware (less than 256KB of RAM)(Warden, 2018).

3.3. **TinyML for Expanding Access.** The most difficult task in expanding applied-ML access is making low-cost hardware available anywhere. Cloud-ML technologies cost thousands of dollars, and their physical power, scale, and operational requirements limit their accessibility. Mobile-ML devices are more affordable and pervasive, but their availability is still limited because of network-infrastructure requirements and other factors.

Research shows that although smartphones have become more affordable, their cost remains a barrier in many low- and middle-income countries (LMICs) (Bahia & Suardi, 2020). Statista estimates only 59.5% of the world's population has Internet access, with large offline populations residing in both India and China (Johnson, 2021). According to Pew Research, 76% of individuals in advanced economies have smartphones compared with 45% in emerging economies. Students and teachers in many developing countries lack the resources needed to learn and use traditional ML.

In contrast, TinyML devices are low cost and pervasive. They are readily accessible, enabling hands-on learning anywhere in the world, and their portability eases demonstration of the complete applied-ML workflow in a realistic setting. Furthermore, TinyML applications are more numerous and easier to deploy than mobile- and cloud-ML applications. However, despite the wide availability of tiny devices, there is little material for teaching TinyML (see Figure 4). The number of general-ML courses far exceeds the number of TinyML courses (or, more generally, embedded-ML courses).

4. An Applied-TinyML Specialization

We developed an applied-ML course specialization focusing on TinyML. Our specialization provides multiple on-ramps to enable a diverse learner population. Moreover, because TinyML is easy to deploy on hardware and test in the real world, it allows us to systematically explore applied ML's vast design space (algorithms, optimization techniques, etc.). It also lets us incorporate responsible AI in all four ML stages: design, development, deployment, and management at scale, which we

[3]For the specific task of keyword spotting as an immersive learning experience, 30–40 samples of 2-4 keywords should be enough to train a model from scratch to do a fairly decent job of disambiguating those few words for one speaker. We do not mean to suggest that this would suffice for a production system.

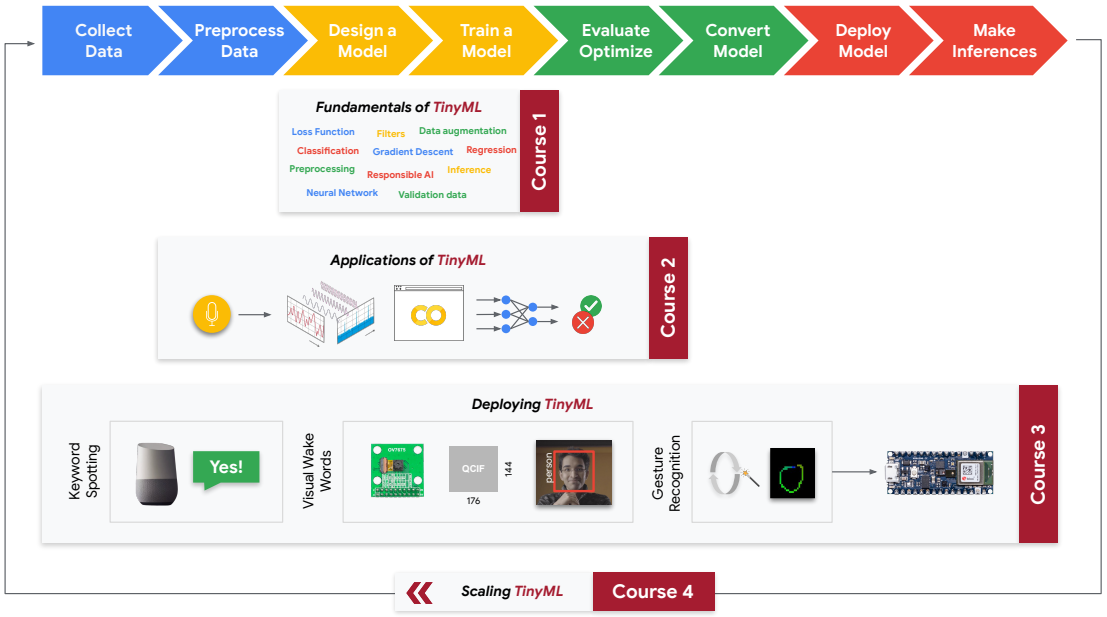


Figure 5. The ML workflow from data collection to model training to inference. The spiral course design focuses on the neural-network model in Course 1, model application in Course 2, application deployment in Course 3, and, finally, Course 4 "closes the loop" by covering ML operations (MLOps) which enable the scaled management, deployment, and continual improvement of TinyML applications.

discuss in greater depth in Section 5. We hope our description of this applied-ML specialization serves as a roadmap for anyone wishing to adopt the program.

4.1. **A Four-Course Spiral Design.** The TinyML specialization comprises three foundational courses and one advanced course, which we consider optional. Participants would ideally start with the first course and work through the natural progression, but we allow them to go in any order they choose. Depending on their background, they can skip some courses and take the one most relevant to their knowledge and expertise.

We structured our course using a spiral design (Bruner, 2009) in which key concepts are presented repeatedly, but with increased complexity as students progresses through the courses. This design allows us to continuously reinforce key concepts, provide multiple on-ramps to the specialization depending on student background, and support well scaffolded hands-on exercises throughout the specialization. As we mentioned earlier, our application-focused spiral design covers the complete ML workflow, going outward from the middle. The curriculum begins with neural networks for TinyML in Course 1, expands to cover the details of TinyML applications in Course 2, deploys full TinyML applications in Course 3, and application management and scaled deployment in Course 4 (Figure 5). Our application focus increases learner engagement and enthusiasm (Yang, 2017) and enables students to not only learn core concepts but also create their own TinyML applications and deploy them onto physical microcontrollers.

Table 2 shows a breakdown of the courses. Roughly, each one takes five or six weeks to complete. For a more detailed and up-to-date overview and links to all materials, visit our open-source courseware at https://github.com/tinyMLx/courseware.

4.2. **Fundamentals of TinyML (Course 1).** Course 1 is titled Fundamentals of TinyML. Its objective is to ensure students understand the "language" of (tiny) ML so they can dive into future

Course 1: Fundamentals of TinyML	Course 2: Applications of TinyML	Course 3: Deploying TinyML
1.1. Course 1 Overview	2.1. Course 2 Overview	3.1. Course 3 Overview
1.2. The Future of ML Is Tiny and Bright	2.2. AI Life Cycle and ML Workflow	3.2. Getting Started
1.3. Tiny Machine Learning Challenges	2.3. ML on Mobile and Edge Devices (1)	3.3. Embedded Hardware and Software
1.4. Getting Started With ML	2.4. ML on Mobile and Edge Devices (2)	3.4. TensorFlow Lite Micro
1.5. The ML Paradigm	2.5. Keyword Spotting (KWS)	3.5. Deploying Keyword Spotting
1.6. The Elements of Deep Learning	2.6. Data Engineering	3.6. KWS Custom-Data-Set Engineering
1.7. Exploring ML Scenarios	2.7. Visual Wake Words (VWW)	3.7. Deploying Visual Wake Words
1.8. Building a Computer-Vision Model	2.8. Anomaly Detection	3.8. Gesturing Magic Wand
1.9. Responsible AI Design	2.9. Responsible AI Development	3.9. Responsible AI Deployment
1.10. Summary	2.10. Summary	3.10. Summary

Course 4: MLOps for Scaling TinyML		
4.1. Course 4 Overview	4.5. Continuous Training	4.9. Continuous Monitoring
4.2. MLOps: The Big Picture	4.6. Model Conversion	4.10. Data & Model Management
4.3. ML Development	4.7. Model Deployment	4.11. Responsible AI Management
4.4. Training Operationalization	4.8. Prediction Serving	4.12. Summary

Table 2. A breakdown of topics in the four TinyML courses. Each one has several activities, including videos, colabs, hands-on labs, quizzes, readings, case studies, assignments, tests, and discussion-forum participation.

courses. TinyML differs from mainstream (e.g., cloud-based) ML in that it requires not only software expertise but also embedded-hardware expertise. It sits at the intersection of embedded-ML applications, algorithms, hardware, and software, so we cover each of these topics. As Figure 5 shows, the course focuses on a portion of the complete ML workflow. Moving to subsequent courses, we progressively expand participants' understanding of the rest of that workflow.

The course introduces students to basic concepts of embedded systems (e.g., latency, memory, embedded operating systems, and software libraries) and ML (e.g., gradient descent and convolution). The first portion emphasizes the relevance of embedded systems to TinyML. It describes embedded-system concepts through the lens of TinyML, exploring the memory, latency, and portability tradeoffs of deploying ML models in resource-constrained devices versus deploying them in cloud- and mobile-based systems.

The second portion goes deeper by focusing on the theory and practice of ML and deep learning, ensuring all students gain the requisite ML knowledge necessary for later courses. Students explore central ML concepts through hands-on coding exercises, training their ML models to perform classification using Python and the TensorFlow library in Google's Colaboratory programming environment. To stay neutral, lessons that students learn and practice are broadly applicable outside of TensorFlow and the free Google ecosystem we leverage. For instance, the fundamentals students learn with TensorFlow are also relevant to other popular frameworks like PyTorch. The only change that students have to adapt to is the application programming interfaces (APIs).

We provide an overview of embedded systems and ML to ensure students recognize that the topics we cover in the specialization are relevant to their lives and careers, boosting motivation and retention (Dyrberg & Holmegaard, 2019; Wladis et al., 2014). For those with sufficient ML and embedded-systems experience, Course 1 is optional. By designing the series with these multiple on-ramps, we can meet participants wherever they are, regardless of their background and expertise.

4.3. **Applications of TinyML (Course 2).** The objective of the second course is to give learners the opportunity to see practical (tiny) ML applications. Nearly all such applications differ from traditional ML because TinyML is all about real-time processing of time-series data that comes directly from sensors. As Figure 5 shows, we help students understand the complete end-to-end ML workflow by including additional stages, such as data preprocessing and model optimization.

Moreover, when we revisit the same stages (e.g., model design and training), we employ spiral design to broach advanced concepts that build on Course 1.

Course 2 examines ML applications in embedded devices. Participants study the code behind common TinyML use cases, such as keyword spotting (e.g., "OK Google"), in addition to how such front-end, user-facing, technologies integrate with more-complex smartphone functions, such as natural-language processing (NLP). They also examine other industry applications and full-stack topics, including visual wake words, anomaly detection, data-set engineering, and responsible AI.

We take an application-driven approach to teaching the technical components. For example, we use the keyword-spotting (KWS) example to demonstrate the importance of preprocessing sensor inputs, showing the power of Fast Fourier transforms (Brigham, 1988) and Mel-frequency cepstral coefficients (Tiwari, 2010) through coding exercises. We additionally explore the importance of holistic architecture by discussing the quality of service metrics that evaluate KWS applications and the "cascade architecture" (i.e., ML models staged one after another for efficiency) for deploying them (Gruenstein et al., 2017). As another example, through the lens of the visual wake words (VWW) application, we introduce transfer learning (Yosinski et al., 2014), teaching students to develop their neural-network models without voluminous training data and expensive hardware. Supplementing the theoretical concepts is a coding exercise that employs transfer learning on a pretrained MobileNet (Howard et al., 2017) model to detect whether an individual is wearing a mask—a real-world application that will resonate with learners in light of Covid-19. As a final example, we use anomaly detection (AD), in the context of predictive maintenance for manufacturing, to demonstrate the power (and limitations) of supervised learning and deep neural networks by exploring k nearest neighbors (G. Guo et al., 2003), an unsupervised traditional-ML technique, and comparing it with autoencoders (Baldi, 2012), an unsupervised neural-network technique.

The course not only teaches students about TinyML applications and their technical components, but also how to run and test these applications using TensorFlow Lite in Google's Colaboratory programming environment. This step completes the second learning spiral, providing a hands-on opportunity to explore full TinyML applications. The inclusion of TensorFlow Lite lets participants explore important TinyML topics (e.g., neural-network quantization), preparing them to add the next layer in Course 3: physical hardware. We intentionally avoided introducing microcontroller hardware until the third course so students could complete the first two entirely for free. They can then make an informed decision on whether to buy the low-cost hardware that Course 3 requires.

4.4. **Deploying TinyML (Course 3).** Most instructional ML material focuses on models and algorithms, failing to provide hands-on experience in gathering input or training data, making decisions on the basis of a model's output, and testing models in the real world. Therefore, Course 3 explicitly focuses on demonstrating the complete ML workflow (Figure 5).

We have found that learners are excited about using the knowledge they gain to solve real problems. But absent guidance in how to build an entire system, many become frustrated because they are unable to apply their knowledge. This issue arises in the form of questions that traditional courses leave out, such as "How can I find the training data for my problem?" and "What threshold should I use to decide whether a classification score is high enough for my application?" and "How do I go from an RGB-camera-image byte array to the float-tensor image my model needs?"

Providing universally applicable answers to all such questions is impossible. But alerting early learners to them and offering a set of comparable guided practical experiences reduces the frustration before these questions arise in their projects, hobbies, or careers. This is critical, as frustration squelches the desire to master the skills a field requires. Instead, we want to give students the support and confidence they need to overcome challenges and solve problems.

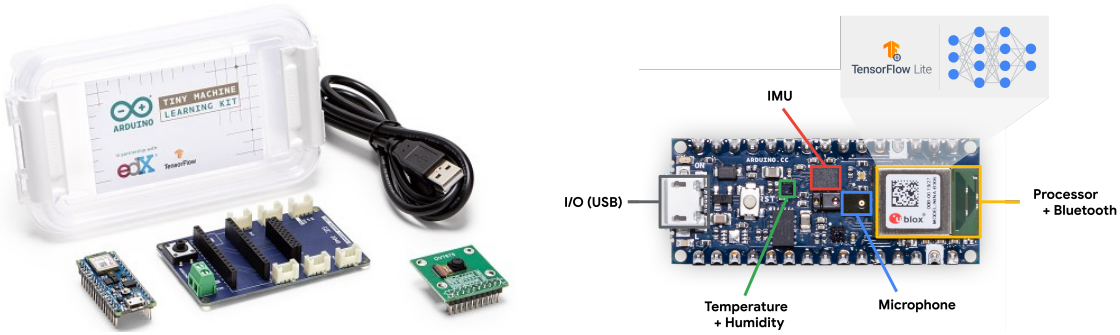


Figure 6. Shown at left, the TinyML Kit includes the Arduino Nano 33 BLE Sense ("Arduino Nano 33 BLE", 2021), an OV7675 camera module ("OV7675", 2021), and a TinyML shield that simplifies sensor integration. Shown at right, the Nano 33 BLE includes a host of onboard integrated sensors (e.g., temperature/humidity, IMU, and microphone), a Bluetooth Low Energy (BLE) module, and an Arm Cortex-M microcontroller that can run neural-network models using TensorFlow Lite for Microcontrollers.

"Deploying TinyML" mixes computer science and electrical engineering. It gives participants fundamental knowledge and hands-on experiences with ML training, inference, and deployment on embedded devices. Following the spiral pattern, it builds atop many of the techniques and applications from previous courses and adds new technical topics and extensions. Students develop and deploy full applications, such as KWS, person detection, audio/visual reaction, and gesture detection on their own microcontrollers.

The course introduces TensorFlow Lite for Microcontrollers (David et al., 2020), an embedded-ML software library that eases the task of efficiently running ML models on embedded hardware. Students learn how the library works, helping them appreciate the challenges that an embedded-ML-framework engineer faces in the real world. They also examine the library's APIs as they deploy applications such as KWS, VWW, and magic wand to their microcontrollers.

Building on the concepts from the first two courses, we introduce new concepts such as multitenancy—that is, running more than one ML model at a time—when we revisit certain stages of the ML workflow (Figure 5). We present the tradeoffs between using multimodal learning that fuses sensor data versus using two separate models to make inferences. The former stresses the first half of the ML workflow (training), while the latter stresses the second half (deployment).

A unique benefit of this course is the exercises that involve the entire ML process. Before they know it, students are implementing an entire TinyML application from beginning to end on a physical device they can hold in their hands. This approach gives our course the unique value of allowing participants to fully develop and use their own TinyML projects at home. This type of hands-on project-based learning is proven to enhance learning, motivation, and retention (Krajcik & Blumenfeld, 2006; Vesikivi et al., 2020). For instance, participants collect their own custom keyword data for training a KWS model, giving them first-hand experience with the challenges of getting ML models to work accurately. Some find that the *tinyConv* (TensorFlow, 2021) KWS model works well; others find that they must collect more data or adjust the preprocessing. A few individuals in this latter category are perplexed that even those improvements may fail to dramatically increase accuracy, finding that the 16 KB KWS model is too small. The point of the exercises is not necessarily to increase model accuracy, but to instead understand the challenges of applying ML models to the real world.

The course uses the Tiny Machine Learning Kit, which we co-designed with Arduino for hands-on, low-cost, accessible, project-based learning. This kit, shown in Figure 6, and described in more detail

in Section 4.7, is globally accessible, and includes an Arduino board containing numerous sensors (microphone, temperature, humidity, pressure, vibration, orientation, color, brightness, proximity, gesture, etc.) and an Arm Cortex-M-class microcontroller (Martin, 2016) which enables a wide range of TinyML applications from image recognition, to audio processing, to gesture detection.

After completing Courses 1, 2, and 3, students are eligible to receive the HarvardX/edX Tiny Machine Learning Certificate, testifying they are trained as full-stack TinyML engineers. We offer the certificate because many professional learners desire such awards to enhance their resumes and prove to potential employers that they have mastered particular skills. At this point in the series, participants have not only explored the technical and societal challenges that TinyML poses, but they have also gained hands-on experience with the complete TinyML-application pipeline: collecting data, developing and training models in the cloud using TensorFlow, testing them in the cloud using TensorFlow Lite, and deploying them in hardware with TensorFlow Lite for Microcontrollers.

4.5. **Scaling TinyML with MLOps (Course 4).** Course 4 is optional, and students can take it after completing Courses 1–3, or it can be taken directly. The first three courses introduce students to TinyML application design, development, and deployment on a device. Course 4 expands on this foundation by examining scaled management of TinyML-application deployments via Machine Learning Operations (MLOps) to address a critical situation—only one in every two organizations manage to move beyond pilots and proofs of concept (Algorithmia, 2020; Capgemini Worldwide, 2020). Models do not make it into production, and if they do, they break due to the inability to respond to changes in the environment. One of the primary reasons is that many teams, organizations, and individuals fail to recognize the need for structured ML processes. It is becoming increasingly clear that to be successful with ML, an automated and optimized ML process workflow is required. The workflow assists teams in putting ML models into production with confidence that they will succeed. It also aids in risk management when firms deploy their ML application(s) to many devices in rapidly changing environments prone to model concept and data drift.

Course 4 focuses on two major aspects of systematically scaling TinyML into the real world. The first half of the course is dedicated to "scaling up" the effectiveness of TinyML applications. Sample topics include benchmarking, model optimization, and automated co-design (e.g. NAS (Zoph et al., 2018) and AutoML ("Cloud AutoML Custom Machine Learning Models | Google Cloud", 2021)). The latter half focuses on "scaling out" TinyML applications from one to thousands of devices, where, for instance, we highlight the issues that embedded systems pose for scale-out deployment. We discuss and address the lack of embedded container-orchestration solutions, such as Docker and Kubernetes, for automating TinyML application deployment, scaling, and administration.

As there are numerous challenges for scaling TinyML, rather than discussing the scale-up and scale-out challenges piecemeal, we instead show learners how to operationalize for scale-up and scale-out methodically using MLOps (Figure 7). Broadly, MLOps is defined as a set of techniques to develop, deploy, and maintain ML models in production reliably and efficiently. In Figure 7, the ML development stage covers experimenting with and establishing a stable and reproducible model training approach (training pipeline code), which includes a variety of tasks ranging from data preparation to model training and evaluation on-device. Much of this is already discussed in Course 1–3, so here we focus more on the personnel involved (Figure 8, discussed in detail later) and introduce the importance of model and data versioning for repeatable experimentation. The ML training stage is where we emphasize the need to operationalize the process of packaging, testing, and deploying training pipelines to be robust. We discuss concepts such as smoke testing and various staging deployment methods, including blue-green and canary deployments, so that learners understand how to seamlessly and safely push models into production. The continuous

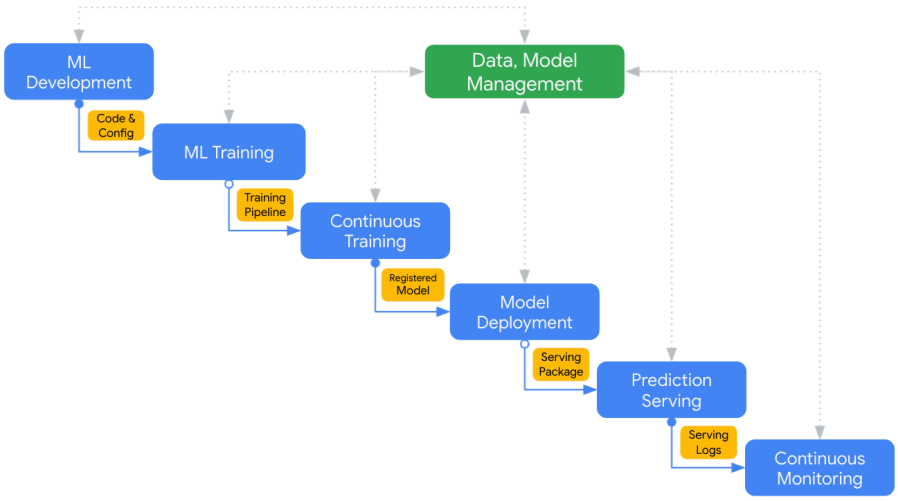
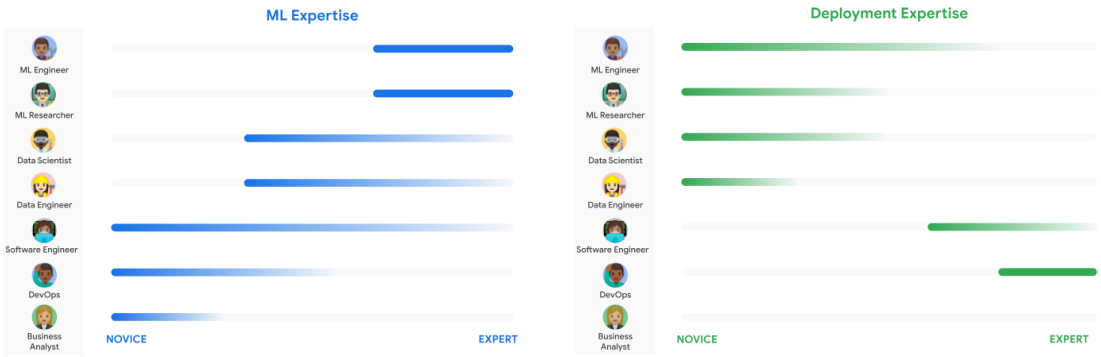


Figure 7. Scaling TinyML using MLOps. We present a systematic approach for developing and deploying models into production. The MLOps stages are shown in a waterfall design, linearly cascading down from one to another, which makes it easy to convey the important aspects within each stage. However, in practice, many organizations adopt an agile, iterative approach because there are often strong interdependencies that necessitate an agile flow.

training stage is about rerunning the training pipeline in response to new data or code changes, or on a schedule, maybe with new training settings. This is where we bring in the need for having downstream triggers in the continuous monitoring stage. We convey that developers need to be proactively informed about when we should update models in response to real-world changes rather than reacting to catastrophic failures in the field that are often too late to fix. The model deployment stage is about the packaging, testing, and deployment of a model to a serving environment for online experimentation and production serving. This is also where we emphasize the importance of understanding the end-user device's capability and developing automated methods for automatically optimizing the models to perform as expected in terms of service response times, energy consumption etc. The prediction serving stage is about serving the model that is deployed in production for inference under various latency and resource efficiency constraints. It is where we introduce some of the embedded system elements that make it challenging to easily and readily deploy "TinyML as a service (TinyMLaaS)." Traditionally, machine learning as a service (MLaaS) eases ML deployment for clients without incurring the associated cost, time, and risk of building an in-house ML pipeline and a team to manage it all. But doing this for TinyML introduces a very unique set of challenges for which there are still no publicly available tools. Nonetheless, we discuss it to present our learners with an opportunity for creating such new services that are needed. The purpose of the continuous monitoring stage is to assess the effectiveness and efficiency of a deployed model. When there are shifts in the concept or model, we need retriggers that automatically initiate the ML (re-)training stage. Finally, the data and model management stage is a critical cross-cutting function that governs ML artifacts for auditability, traceability, and compliance. Data and model management can also help ML assets be more shareable, reusable, and discoverable.

A significant takeaway from Course 4 is that constructing an ML-enabled system at scale requires numerous personnel, including ML engineers and researchers, data scientists and engineers, software developers, an IT operations team, and business analysts who establish important performance indicators, to practice MLOps (Figure 8). Each of these contributors' expertise will likely vary. For instance, an ML engineer or researcher may be an expert at developing machine learning models (Figure 8a). However, they will mostly likely lack the necessary expertise to push models into



(A) ML expertise. (B) Deployment expertise.

Figure 8. ML personas. Many different people are involved to realize ML-enabled applications in the real-world. It is important to understand this big picture to appreciate the role that everyone places in a successful ML use case.

embedded deployment for production (Figure 8b), and vice versa. We want learners to understand the big picture and realize that ML, in practice, is more than just ML; it is an ensemble of experts.

In summary, Course 4 concludes the TinyML series by defining the essential procedures and technical capabilities in an MLOps framework, so that learners from Course 1–3 understand how to take their new found TinyML knowledge and put it into practice in the real-world. Those who take only Course 4, benefit from understanding the role that various personas in MLOps play and how to implement a practial workflow. The methodology, which we adopted from the industry, can help organizations construct mature MLOps strategies for building and operationalizing TinyML systems. Adopting a systematic framework, be it the one we discuss or some other, can assist enterprises in improving team cooperation, increasing the reliability and scalability of TinyML and shortening development cycle times. These advantages, in turn, drive innovation and contribute to the overall business value of TinyML. Participants who finish all four courses will have studied the essentials of ML-model creation, development, deployment, and management through the TinyML lens. This expertise is crucial for advancing one's career in this rapidly developing industry.

4.6. **Student Activities.** People learn differently (Pashler et al., 2008). To support many learning styles, we implemented proven strategies (Lockman & Schirmer, 2020) and a variety of methods (Figure 9). Our approach mixes video lectures, short readings, and coding exercises in Google's Colaboratory programming environment to teach and reinforce the course's main technical components. Thus, visual, auditory, and experiential participants all learn by their preferred method.

We keep the videos short (4–10 minutes). Research shows that people learn better from numerous short content modules than from a few long ones (P. J. Guo et al., 2014). Because some students will find that many of the coding components are new, we provide walk-throughs of major sections, numerous comments in the code, and introductory text to explain the purpose of each code snippet.

In the first two courses, each section builds toward a coding assignment to encourage project-based exploration and creativity. The assignments in Course 3 expand to full-on hardware deployment that lets students hold their own designed, trained, and deployed model in their hands and test it in the real world. The activities grow in complexity and detail as students progress through the courses, following our spiral-design principle. Students thus gain confidence throughout, as complete application deployments can be challenging. Finally, we sought the input of industry veterans on our course staff to ensure the hands-on activities build relevant full-stack skills. As

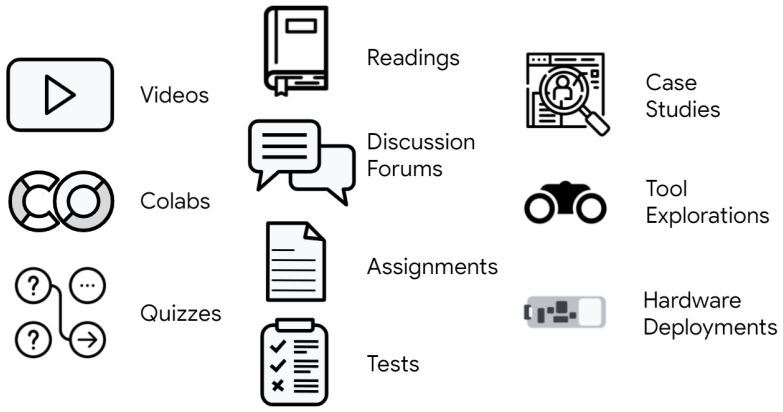


Figure 9. We employ a wide array of learning activities to give students an immersive, self-paced online experience.

such, in Course 4, we also added in tool explorations and case studies to further demonstrate to students, and give them practice understanding, how to leverage TinyML at scale in the real world.

The courses also include formative multiple-choice quizzes throughout, focusing on the main concepts so students see their progress even if they do not understand every line of code. The quizzes also reinforce the importance of high-level tradeoffs and applied-ML concepts, which will be relevant to ML careers even if the technical stack changes. For those pursuing a paid certificate, we also included summary tests at the end of each section.

Finally, we provide many discussion forums that allow students to ask questions and get answers. Forums allow the course staff to support all participants regardless of their location. They also serve the dual function of building a community around the course. Our forums encourage students to ask any and all questions and to answer them for one another.

Each activity includes a strong ethics component, which we describe in detail later (Section 5). Briefly, however, we ask many open-ended questions to elicit student opinions on the opportunities and challenges of responsible TinyML-application design, development, and deployment. As the literature predicts (Lockman & Schirmer, 2020), many of these questions have led to conversations and debates between our online participants, despite their different geographic locations, ages, and technical backgrounds.

4.7. **Accessible, Hands-on Learning.** To enable hands-on learning anywhere in the world, we need a low-cost, self-contained yet extensible, approachable yet representative, and globally accessible TinyML platform. Once again, microcontrollers are promising because they are inexpensive and widely available. To provide an easily accessible out-of-the-box experience, we designed the Tiny Machine Learning Kit (Figure 6) with Arduino. This section describes the kit and its development.

Systematic selection. The range of TinyML hardware and software options is wide. We searched for one that not only made it simple to integrate the sensors required for the course but also supported easy integration of additional sensors for future study.

Putting application-level software aside leaves two fundamental elements: hardware, from the microprocessor to peripherals and discrete circuitry, and software, from the application layer (our focus) to the silicon layer. An initial constraint on the field of potential microprocessors is the need to support TensorFlow Lite for Microcontrollers (David et al., 2020), which is written in C++ and requires that the microcontroller support 32-bit computing.

We developed criteria for compatible microcontroller development boards, recognizing that an integrated off-the-shelf product would greatly increase accessibility. These criteria include a small

form factor (it is *tiny* ML, after all), a low power budget (efficiency is critical to edge computing), a small system memory (some controllers have large memories, making them less accessible and limiting their range of application), sufficient clock speeds, wireless-communication capability (to enable periodic reporting and/or distributed systems), select sensor integration, and serial channels for extensibility. We defined similar criteria for the accompanying software, comprising the development environment, embedded framework, and logistics (fast, reliable distribution). Next, we added weights to the selection criteria and compiled the candidates in a Pugh matrix (Cervone, 2009). We ranked a field of about two dozen hardware products, giving some preference to controllers that had undergone more-extensive testing—in particular, Arm's Cortex-M series (Martin, 2016) and Espressif Systems' products (namely, the ESP32t) (Systems, 2021). Both of these embedded systems are widely popular.

The TinyML kit. Ultimately we selected the Arduino Nano 33 BLE Sense ("Arduino Nano 33 BLE", 2021) because it uniquely blends expert embedded-systems engineering and remarkable isolation of the application developer from many low-level hardware details (Jamieson, 2011). Furthermore, the Arduino framework and its software APIs ("cores") fit naturally with our spiral design. Arduino's many libraries and simple IDE are easy for inexperienced students to learn, yet it typically permits those interested in the "bare metal" to work their way down the embedded-software stack. Moreover, the Nordic nRF52840 Cortex-M4-based controller ("nRF52840", 2021) on the Nano 33 BLE Sense development board, along with its Mbed real-time OS ("Mbed OS", 2021), represent industry-level hardware and software.

We also developed the Tiny Machine Learning Shield to enable plug-and-play integration of sensors that the Nano 33 BLE Sense lacks. In particular, it eliminates the need for users to make 18 individual connections between the microcontroller and the low-cost camera module we selected for the course—the OV7675 ("OV7675", 2021), which typically sells for about US$2. A series of Grove connectors (Schubert et al., 2013) line each side of the shield for connection to numerous additional sensors, which students can purchase for their own projects and integrate without soldering or low-level circuit design.

We bundled the Nano 33 BLE Sense with the shield, the OV7675 camera module, and a USB cable to form the Tiny Machine Learning Kit (Figure 6); learners can purchase a single item and be fully prepared for Course 3 for US$49.99. To accommodate those few who prefer to purchase elements individually, we provide wiring diagrams and a custom Arduino software library so they can readily swap the OV7675 for the related OV7670 camera module.

Alternatives. In the months after we developed and announced our TinyML kit, similar boards emerged to provide alternative options. For example, the Pico4ML by ArduCam ("Pico4ML by Arducam", 2021) is a notable single-board example that comes complete with a microphone, inertial measurement unit (IMU), and camera module, and is suitable for the course exercises. We are working to support some of these new and exciting hardware platforms to give students more flexibility with their projects.

5. Integrated Ethical & Responsible AI

Ethical and Responsible AI is about putting people, social benefit, and safety first. More specifically, ethical AI emphasizes the need for ML engineers to safeguard user privacy and security, mitigate algorithmic bias and discrimination, and ensure ML models perform reliably after deployment. It also extends to developing consumer trust. In this section, our goal is to shift learners from thinking about which ML technology is feasible to which is useful, with an understanding of how it will influence users and society.

5.1. **Ethical Consideration of Ubiquitous ML.** TinyML offers many helpful features, ranging from data privacy and security to low latency and high availability. Coupled with low-cost embedded hardware, these features make it a pervasive technology that can enable ML everywhere. TinyML sensors will monitor the environment in which they are deployed, be it mechanical or human, around the clock. With the prospect of ML everywhere comes a pressing need to address privacy, drift, bias, and other ethical issues.

Fortunately, TinyML allows us to incorporate responsible AI into all four ML stages: design (Course 1), development (Course 2), deployment (Course 3), and scaling (Course 4). By embedding ethics into each TinyML course, we communicate the technology's ethical and social dimensions in a personal and practical manner.

To achieve deep integration, we follow the Embedded EthiCS pedagogy at Harvard (Grosz et al., 2019), where philosophers participate directly in computer-science courses to teach students how to think through the ethical and social implications of their work. We collaborated with a philosopher from this program to co-develop and include such material in our curriculum. Her commitment to learning the technical aspects of TinyML enabled us to customize the ethical content to meet the unique course needs of TinyML.

By distributing responsible AI throughout the series, covering the entire ML workflow, students discover how ethical issues permeate all aspects of their work. Our aim is to introduce them to the conceptual tools for navigating these issues, in hopes they will view responsible AI as an active enterprise. Next, we describe our pedagogical goals for each responsible-AI unit, some examples we covered, and the exercises that reinforce the concepts.

5.2. **Designing AI Responsibly (Course 1).** Access to, adoption of, and use of ML products is inequitably distributed. According to Pew Research, 64% of Americans believe technology companies create products and services that benefit people who are already advantaged, and 65% believe these companies fail to anticipate the societal impact of those offerings (Smith, 2018).

To enable more-widespread, safer, and more-secure ML, we must raise awareness of its capabilities. Thanks to the low cost and accessibility of TinyML hardware, our students are diverse, and they will probably have to address different social and cultural factors when designing ML applications. To ensure they all can anticipate the effects of ML products and ensure equitable access, our approach to responsible AI focuses on forming a vision of both the problem to be solved and the people a solution will affect.

We believe that by taking an active role in responsible ML design, students will be better able to address ethical challenges such as bias, fairness, and security. We therefore cover real-world examples, such as a Winterlight Labs auditory test for Alzheimer's disease. In this case, research revealed that nonnative English speakers are more likely to be mistakenly flagged as having Alzheimer's (Fraser et al., 2016). In a discussion forum, students reflected on what the product designers could have done differently to avoid this failure. Such activities reinforce the importance of considering diverse user perspectives during the design phase, as doing so can inform data-collection decisions that mitigate ML bias.

In a subsequent forum, learners practice ethical reasoning about the consequences of a KWS model's failure in terms of Type I (false positive) and Type II (false negative) errors. In this case, a false positive would result in audio being recorded, unbeknownst to the user, and sent to the cloud. A false negative means the device failed to activate when the user spoke the wake word. Students must justify their decision to optimize the model for high precision, thereby minimizing false positives, or to optimize for high recall, thereby minimizing false negatives.

For the KWS activity, nearly all participants chose to optimize for high precision to minimize the risk of privacy violations. Interestingly, one provided a justification based on sustainability concerns related to unnecessary data transmission and storage in the cloud. Those who decided to optimize for high recall cited a variety of reasons. One noted that although people claim to value privacy, they tend to prioritize convenience. In contrast, another suggested enacting privacy measures elsewhere to offset the potential harm of optimizing for high recall. Lastly, yet another student prioritized model performance to meet user expectations. That student claimed the burden of preserving privacy should fall on the user, who has the ability to decide whether to purchase the product. There is no right or wrong answer. Our desire is to spur self-reflection and foster constructive discussion among learners from different backgrounds and cultures.

5.3. **Developing AI Responsibly (Course 2).** Any developer employing ML must be aware of how data-collection bias and fairness affect application behavior. Our courses use public data sets, including Speech Commands (Warden, 2018), Mozilla Common Voice (Ardila et al., 2020), ImageNet (Deng et al., 2009), and Visual Wake Words (Chowdhery et al., 2019), for nearly all of the programming assignments. Most data sets, however, have demographic-representation problems (Buolamwini & Gebru, 2018). For example, despite crowdsourcing efforts to increase diversity, the Common Voice data set lacks equal gender representation (only 24% of English-data-set contributors who revealed their gender are female) ("Discourse Discussion Forum: How will the lack of female voices be handled", 2019).

Our goal is for students to see how data collection, bias, and fairness intertwine, as well as to equip them to mitigate the problems. because they are working with KWS models, we cover real-life biases relevant to this kind of ML application. For instance, research shows that voice-recognition tools struggle to identify African American Vernacular English, causing popular voice assistants to work less well for black individuals (Koenecke et al., 2020). Similarly, research shows that voice recognition struggles to identify nonnative English speakers and those with speech impairments (Wu et al., 2020). To acquaint learners with recent work in mitigating bias, we discuss Project Euphonia, a initiative that launched in 2019 with the goal of collecting more data from individuals with speech impairments or heavy accents to fill the gaps in voice data sets (Shor et al., 2019).

We created a Colab activity that uses Google's What-If Tool (WIT) (Google, 2020b), based on its Responsible AI tool kit (Google, 2020a). The WIT is one of the company's many open-source, interactive visualization tools for investigating trained-ML-model behavior with minimal coding. In this exercise, participants practiced ethical reasoning by exploring a real-life data set, identifying sources of bias, and evaluating threshold-optimization strategies for fairness. In general, learners responded very positively to the WIT module in the subsequent forum. Some students noted how the visual representations fostered a deeper understanding of issues pertaining to fairness. One claimed that the focus on confusion matrices in particular was an effective way to clearly distinguish between the fairness metrics.

5.4. **Deploying AI Responsibly (Course 3).** Even after designing and developing an ML model, deployment raises a new set of ethical challenges. For example, TinyML systems are often touted for preserving privacy. When an embedded system processes data locally (close to the sensor) rather than transmitting it to the cloud, we tend to believe it protects user privacy. But user interaction with the model raises new privacy and security concerns (Prabhu et al., 2021). Moreover, ML interacting with a dynamic real-world environment using sensors raises concerns about model drift[4] over a product's lifetime.

[4]*Model drift* generally refers to prediction-accuracy degradation owing to environmental changes.

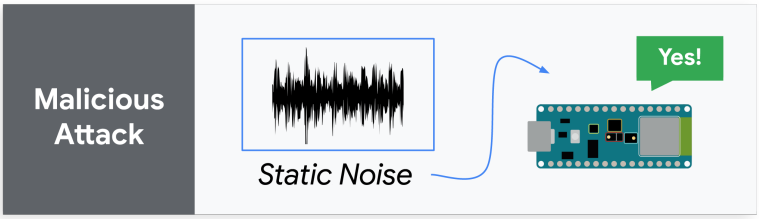


Figure 10. Students attack a pre-trained KWS model with malicious static noise and trigger a spotting of the keyword "yes," showing them the importance of security and privacy.

To the extent that TinyML enables ML everywhere, the privacy, security, and even model-drift risks could be more widespread compared with traditional ML. To familiarize students with these risks, we cover real-life examples, such as doorbells that share data with law enforcement (Harwell, 2019) and fitness devices that leak user information (Whittaker, 2018).

Our goal is to equip students with strategies to mitigate these risks when deploying trained models in embedded devices. Importantly, the mitigation strategies available to traditional ML systems are sometimes unattractive or infeasible for TinyML. For instance, the resource constraints of an embedded device, such as low power and small memories, complicate implementation of robust security systems and model retraining. Therefore, we acquaint course participants with a wide array of strategies, such as minimizing the transmitted and stored data to preserve user privacy, minimizing hardware design to limit vulnerability to attackers, and running supervised experiments in the real world before releasing ML models.

Inspired by research showing we can use inaudible ultrasonic noise to trigger or eavesdrop on KWS models (G. Zhang et al., 2017), we created an exercise that gives students hands-on experience attacking a KWS model. They trigger a false positive—"yes"—with seemingly innocent but adversarial static noise (Figure 10), which in a real application would cause the system to constantly record and transmit the audio. This experience builds on our videos and readings and makes the security threat real—a crucial part of any major security-awareness program at large (Reinheimer et al., 2020). At the same time, it is also a cautionary tale of ML's limitations, a lesson all applied-AI engineers should learn.

To further reinforce this point, a subsequent discussion forum allows course participants to practice ethical reasoning to determine when malicious triggering of a false positive can cause serious harm. Some have noted that this vulnerability would be most likely to cause harm where security is a paramount value, such as using KWS to grant access to a secure space or to initiate a financial transaction. One student drew a connection to the practice of ethical hacking, or penetration testing, and the possibility of developing adversarial data for retraining the model to be more resilient. Interestingly, another noted that since users lack the ability to fix security issues, their only option is to stop using the device. But this choice ultimately depends on whether the company informs customers about the vulnerability. Lastly, one student claimed the adversarial example was more reliable than that student's own voice for triggering a "yes." The course staff then responded, prompting a discussion of data-set bias and the likelihood that American male accents are overrepresented in the data.

5.5. **Scaling AI Responsibly (Course 4).** Many ethical implications require consideration when applying technology. Even minor biases, which can be difficult to detect in the proof of concept, can have a major impact when appearing in thousands or millions of devices.

This problem highlights the need to treat responsible ML as an iterative process. Rather than introduce entirely new ethical considerations, we revisit and expand on previous ones. For instance, to guide students in cleaning up a test data set before they conduct benchmarks, we revisit the ethical issues of data collection and bias. Similarly, we revisit privacy in depth once participants become acquainted with federated learning.

We are incorporating an active learning exercise using Google's Model Card Toolkit (MCT) (M. Mitchell et al., 2019). Model cards are a reporting mechanism that can increase model transparency and facilitate the exchange of information between model creators, users, and others. This exercise requires that students practice using the model-card framework to document information relating to the model's development, performance, and ethical considerations.

We additionally discuss the environmental impact of large-scale TinyML networks, as the production and maintenance of billions of MCUs can lead to substantial carbon emissions. Beyond the ethical pitfalls of scaling TinyML, we cover the potential positive social impact this technology can have in domains such as environmental sustainability, public health, and AI equity.

6. Improving Accessibility via MOOCs

In this section we describe how we leveraged technology to make the TinyML specialization broadly accessible and highlight important considerations made to support our remote learners.

6.1. **Massive Open Online Course.** Our goal was to reach a global audience. We therefore chose to employ massive-open-online-course (MOOC) platforms. Examples such as edX and Coursera are ideal for making the content globally accessible; students need not travel to a different country to learn. These platforms host a wide variety of university-level courses and are generally cheaper than equivalent academic and professional training thanks to the economics of scale (Belleflamme & Jacqmin, 2014). We deployed the TinyML specialization on edX through HarvardX.

Participants can audit the course for free or pay to earn a professional certificate. Since they can "upgrade" to a professional certificate at any point during the course, both students and professionals can try before they buy, encouraging more to enroll. Although the professional certification includes summary tests that are absent from the audit version, we designed the curriculum so individuals who are just auditing learn the same crucial principles. Thus, all can attend the entire class, developing their skills for free. At the time of this writing, the number of auditors far outweighs the number of paying students by more than order of magnitude.

The course is asynchronous and self-paced. Students progress through the material at whatever speed they find comfortable. But unlike in-person courses, interaction between students and staff is minimal, forcing staff to develop high-quality, self-explanatory, and self-sufficient materials that rely heavily on media (which we describe in greater detail in Section 6.2).

Unlike most MOOCs, Course 3 employs the TinyML kit (Section 4.7) for hands-on learning. To maximize hardware accessibility, we worked with Arduino to make a custom all-in-one kit globally available for purchase through either that company's website ("Arduino Tiny Machine Learning Kit", 2021) or one of its many distributors. We also provided a detailed bill of materials for students who wish to buy individual components instead. The main benefit of this approach is that it improves the efficiency for the host institutions by reducing the burden on them for managing inventory and shipping logistics (taxes, international shipping rates, etc.).

6.2. **Accelerated Remote Media Production.** The typical development timeline for a series of online courses, such as the ones we described in Section 4, is about two years—far too long to keep up with changing ML technology. Applied ML, especially in the context of TinyML, remains

a nascent yet quickly developing field. Therefore, media production for the online curriculum must be rapid to ensure the material is timely and relevant and to ensure broad access.

We compressed the media-production time greatly, achieving an average development cadence of 6–8 weeks per course. To maintain this cadence, we created a custom remote-media-production workflow. We produced the TinyML course under the specter of Covid-19, but regardless of the safety limitations, a remote production strategy would still have been the only way to achieve these quick results. Remote production methods offer flexibility and allow an international crew to make contributions, meaning the process continues around the clock. Regardless, no matter when and how it is done, creating a flexible workflow requires a principled content-design approach, and advanced technology is necessary for rapid progress. The following breakdown can serve as a roadmap for others attempting to follow a similar approach.

Production design. To expand access such that our effort meets the needs of a global audience (Section 2), we built our media-production strategy around five critical ingredients: compelling instructional narrative, best media practices for online learning, a diverse and skilled production team, prioritized use of production equipment, and willingness to innovate.

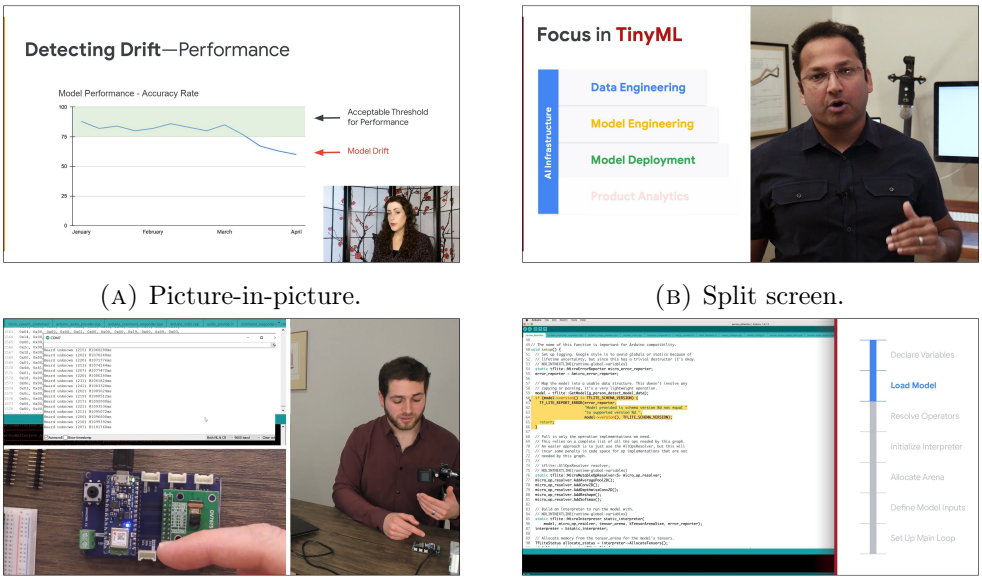
A *compelling instructional narrative* that whets student interest is critical, as all great media experiences unite around a good story. TinyML offers a sound narrative because it provides an accessible, hands-on introduction to ML (Section 3). We aimed to communicate with a global audience and provide the practical knowledge for building complete, relevant TinyML applications.

Effectively communicating that narrative requires *best media practices for online learning*. Decisions made in postproduction often hold more weight than any others. For example, the decision of when to show the instructor, slides, or both in a picture-in-picture cut versus when to display graphics or other visual/auditory information can affect the viewer's cognitive load and overall learning (C.-M. Chen & Wu, 2015; R. E. Mayer et al., 2020). When in doubt, Mayer's "12 principles of multimedia learning" (R. Mayer & Mayer, 2005) is an excellent place to discover such general practices for enhancing the student's experience.

From the start, we determined the primary media types we would produce to hold students' attention and maintain their cognitive load balance (C.-M. Chen & Wu, 2015). We chose picture-in-picture and split-screen formats, allowing us to show the instructor or other imagery in full-screen mode to focus on the most important aspects of the presentation (Figures 11). We emphasized instructor screen time, however, to improve student learning (Wang et al., 2020).

A *geographically distributed and responsive team* is necessary to quickly produce highly sophisticated content, especially for an emerging technical field. Our media team included a producer and director to establish a creative vision and ensure media delivery, a senior editor to assemble and craft the videos, a motion-graphics designer to provide custom graphical elements for our brand, and a production assistant to wrangle data, review content, and integrate the final videos into the platform. This team was relatively lean. One additional advantage was that contributors were scattered across 12 time zones (San Francisco to Boston to London to Mumbai), meaning at all times someone was awake and working on the project.

A crucial ingredient to quickly producing content is *prioritized use of production equipment*. The remote nature of the production and Covid-19 precautions only heightened this need. For example, webcams and audio supplies were sold out or on back order because people were setting up home offices. Fortunately, we were able to make acceptable compromises and buy equipment in a way that ensured the greatest impact. We prioritized production-equipment purchases as follows: 1) audio, because it is more important to retaining viewer attention than video (Canada, 2014); 2) lighting, as it can improve even a nonideal camera to draw the student's eye; and 3) video, which



(A) Picture-in-picture.

(B) Split screen.

(C) Tri-frame for lab demos.

(D) Screencast.

Figure 11. We used various video-production strategies throughout the course to maximize learning efficiency. (a) Picture-in-picture places a video clip in a small frame on top of another frame, playing them simultaneously. It enhances the perception of instructor presence while showing the student relevant content. (b) Split-screen, a slight variation, also improves the perception of instructor presence. (c) Tri-frames are useful for lab demos to enable "hands-on" instruction from teaching assistants. (d) Screencasts coupled with associated lecture material reinforce concepts with code.

we mention last because it is the most expensive in a context where higher production value does not necessarily imply a better learning experience (P. J. Guo et al., 2014).

The final ingredient was *willingness to innovate*. Course 3 (Deploying TinyML) involves hands-on learning. Typically, in-person teaching assistants (TAs) demonstrate labs to show students the goals and scope and to preemptively troubleshoot common errors. Doing the same online is extremely difficult. We developed a three-way split-screen medium (Figure 11c) that displays the device assembly, device testing, and TinyML lab exercises. We assembled a new film location to (remotely) support the teaching staff with the lab exercises, adding an overhead camera and additional lighting. Furthermore, we enhanced our visuals for the three-way split screen with a custom motion-graphics layer. This setup reached completion and underwent rapid testing without disturbing the production timeline. From start to finish, Course 3 took only eight weeks despite involving five hours of produced-video time, which included short lectures, screencasts, and labs.

Technology. Without globally accessible technology and services, remote media production at the level and pace we achieved would have been impossible. Cloud storage was the backbone of our strategy. It allowed contributors to ingest and manage footage globally. It was also the heart of our production workflow, giving us the ability to sync media project files instantly. Videotelephony services such as Zoom and Google Meet aided in assessing home-studio setups in addition to serving as a virtual rehearsal stage and writers' room. Amazon supplied 90% of our equipment. Frame.io streamlined our video-quality review and revision (Frame.io, 2021).

Copyright. Although on-camera presence was a major focus of remote production, video lectures are just one part of the students' activities. At the same time, a multidisciplinary team of content experts, graphic designers, and web developers at HarvardX rapidly designed and formatted

readings and coding exercises. A major challenge in quickly producing course materials was ensuring each illustration, photo, and code library met strict licensing requirements to avoid copyright infringement. Given our project's more than two thousand graphics and tight timelines, we trained all content developers on proper sourcing for course materials. In-house custom graphics—necessary for a nascent field—predominated, and copyright specialists at HarvardX evaluated each piece as it arrived to cite all external creators.

6.3. **Building Community.** A common and well-known pitfall of MOOC platforms is the difficulty of developing community and fostering peer learning among a geographically distributed population. Students often struggle to discuss and collaborate after completing the course and even during the course. We therefore developed the TinyMLx community, which welcomes everyone beyond the edX platform.

First, we created a Discourse forum to provide both a communication platform for students and a home for future initiatives. It has been successful, garnering over 3,500 user visits and over 58,500 page views in its first five months. We also conducted two live Q&A sessions for the TinyML community. For each session, between 100 and 200 learners joined live from around the world, and many more have since watched the recording. We received dozens of questions leading up to the events and dozens more during, with topics including how to best teach TinyML material, how to improve diversity in TinyML, and many others in between. Participants enjoyed the events, e.g., 90% of respondents to our first post-event poll said they would like to attend another. Finally, based on learner feedback we recently created a Discord chat to further enable easy collaboration, communication, and community building.

To challenge our Course 3 students, who went on to deploy TinyML models on their microcontrollers, we developed an optional "capstone-project" competition. We believe this competition reinforces the value and usefulness of the technical skills that students are gaining. A prize will go to the individual (or individuals) whose project demonstrates technical mastery, is most creative in its implementation, and has the most potential to improve society. This initiative has already spawned collaborative-learning groups.

To increase the impact of these projects and further reinforce the real-world applicability of the knowledge students gain through this course series, we are working with the Arribada Initiative ("Arribada Initiative: Open Source Conservation Technology", 2021) to create larger advised projects. This partnership will allow students to contribute their newly acquired TinyML skills to real-world conservation efforts, such as human/elephant conflict mitigation and sea-turtle monitoring, while receiving advice and support from both industry professionals and course staff. Finally, we are asking the community to continuously improve the course, since it is as much theirs as ours. As a result, we've seen many forum posts and GitHub pull requests offering typo corrections, bug fixes, and even content-improvement suggestions.

7. Broader Impact

Our goal is to expand global access to applied ML through the lens of TinyML. In this section, we assess our work's initial impact by presenting data from edX Insights, a service that provides course statistics to instructors and staff. It is merely an initial impact assessment, as the first cohort of participants have just begun graduating from the core TinyML series (Courses 1-3), and Course 4 (optional) remains in development. As such, our early analysis considers enrollment in the first three courses by geography, background, age, and gender.

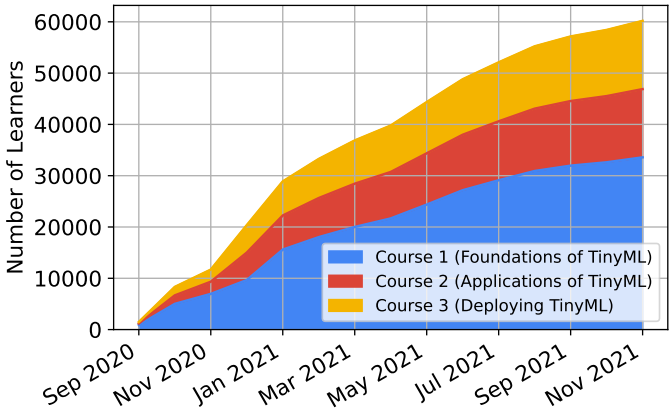


Figure 12. Course-enrollment metrics for Courses 1 through 3. Over 60,000 students are currently enrolled across all three courses. Course 4 has yet to open for enrollment. We expect another enrollment spike with "Scaling TinyML."

7.1. **Course Enrollment.** At the time of this writing, the total course enrollment stands at over 60,000. Figure 12 shows the daily enrollment data, starting from the opening date. We announced Courses 1, 2, and 3 together in early October 2020 and launched them on October 27, 2020; December 16, 2020; and March 2, 2021, respectively. Students could enroll in any or all courses at the same time but could only start after each course's launch date. Course 4 will start next year.

TinyML is a young field, so the first useful metric is interest in the topic (i.e., acquiring applied-ML skills via TinyML). Figure 12 shows the strong and steady increase over time. On average, ∼1,000 new students enroll in at least one course each week. Interest in Courses 2 and 3 continues to grow—a phenomenon we attribute to participants promoting them through social media such as LinkedIn, Twitter, and Facebook as they earn their course-completion certificates. The sharp increases around the first week of October, third week of December, and third week of February align with course-announcement dates or major social-media activity. For instance, on January 24, Mashable handpicked "Fundamentals of TinyML" as one of the 10 best free Harvard courses to learn something new (Turner, 2021). TinyML ranked at the top of the STEM-courses listed.

TinyML students come from more than 176 countries. Because edX reaches a wide audience, our learners come from nearly all continents. Today, the top 10 countries by participant activity are the US, India, Turkey, the UK, Canada, Pakistan, Germany, Brazil, Australia, and Indonesia.

7.2. **Completion Rates.** People take online courses for a wide variety of reasons. Some are curious about the topic and want to get their feet wet; they may audit a course but not complete it. Others would like to master the program and earning a certificate of completion, assuming they can afford it. Therefore, enrollment numbers alone are insufficient.

We assessed how many verified enrollees complete the course. We have access only to the percentage who have earned a passing grade among those officially enrolled in the courses (i.e., the paid-certificate program). This number is constantly changing. At the time of writing, the completion rates are 59%, 55%, and 44% for Courses 1, 2, and 3, respectively. We believe Course 3's number is slightly lower because it is more challenging than Courses 1 and 2, which do not have a hands-on component. We also believe it is lower because of the global chip shortage ("The Global Chip Shortage: A Timeline of Unfortunate Events", n.d.), caused partly by COVID-19, which has affected the TinyML kit availability. The average completion rate for most MOOCs is somewhere between 5% and 15% (Hollands & Kazi, 2019), so the TinyML courses appear to be

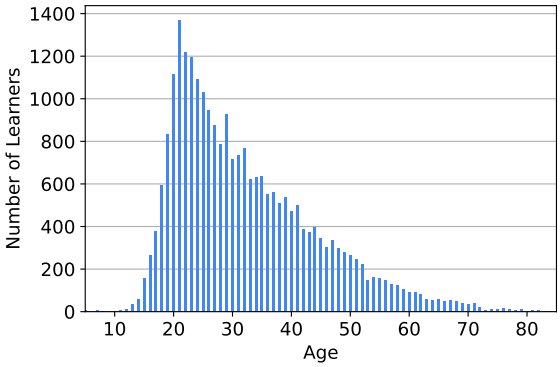


Figure 13. Age demographics across all courses based on voluntarily provided information. We have learners who are still in high school to learners who are retired and learning TinyML to understand its impact on our global society.

faring well. Although these results are preliminary (we need more data to make better quantitative comparisons), we believe that they shed a positive light on our design approach.

7.3. **Learner Demographics.** We conducted a demographic analysis of students' age, educational background, and gender. They volunteer this information to edX, so it covers only a fraction of the numbers in Figure 12. Nonetheless, the data is extensive enough that we can draw general conclusions. At the start of each course, a forum post asks students to introduce themselves and summarize what they hope to get out of the edX series. We derived additional qualitative analysis from these responses. So far we have a good distribution across age groups and educational backgrounds. Our gender diversity is lacking, however, but we are working to address it (Section 8).

Age. Figure 13 shows the age distribution for all three courses combined. The median is 30. Some participants are high-school students as young as 15 and wish to pursue an ML career. Others are over 60 and wish to understand the latest technological innovations as well as their societal implications. This age diversity was one of our objectives (Section 2).

Education. Figure 14 shows that nearly all our learners have either just a secondary (high-school) diploma or a bachelor's/master's degree. A few others have doctorate degrees. Judging from the forum discussions, we gather that individuals with a bachelor's or master's degree are trying to advance or shift their careers by adding an ML focus. Most participants with a doctorate want to apply (tiny) ML in their research. Many students expressed enthusiasm about enrolling in a career-advancing course backed by both Harvard and Google. This variety of educational backgrounds and career focuses also meets our expectations and objectives and further emphasizes the importance of our academia/industry partnership (Section 2).

Gender. Figure 15 depicts the gender diversity across all three courses. It weighs heavily toward men; on average, across all three courses, 20% of our learners are women. We are working to change that ratio by putting together a working group to encourage more women to learn about TinyML.

8. Future Directions

TinyML can dramatically transform applied-ML education and development at many levels, far beyond what we achieved with the edX specialization. To this end, we launched the Tiny Machine Learning Open Education Initiative (TinyMLedu) (Plancher & Janapa Reddi, 2022) to sponsor a wide variety of working groups focused on enabling others to learn about and teach TinyML. These include: TinyML4D which focuses on bringing TinyML materials and resources to developing countries; TinyML4STEM which focuses on nurturing creative research in science,

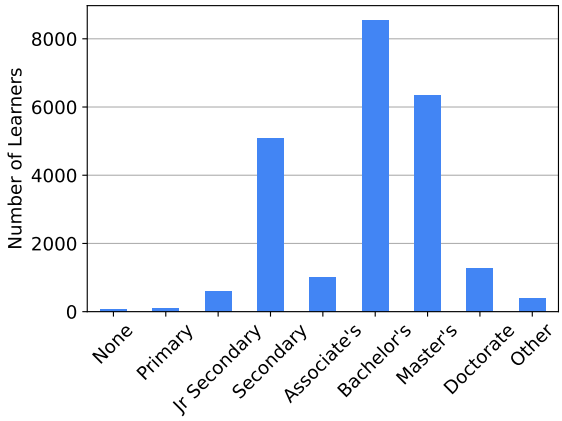


Figure 14. Education demographics based on voluntarily provided information. Many of our learners indicated an interest in TinyML to understand applied ML technologies to either pivot or grow further in their current positions.

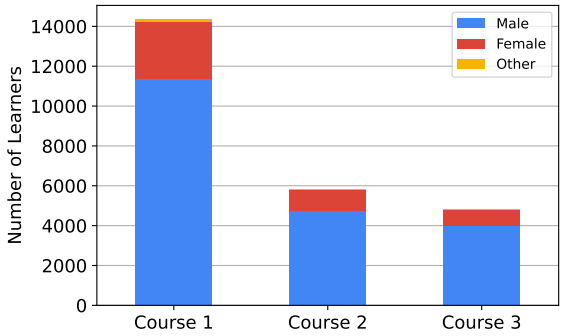


Figure 15. Gender demographics based on voluntarily provided information. Collectively, we are working on engaging a more diverse population of learners with the aid of working groups that are part of the tinyML Foundation.

technology, engineering, and math; TinyML4K12 which focuses on integrating ML and TinyML into the K12 classroom; TinyML4Xlation which focuses on translating TinyML materials into languages other than English. In particular, since MOOC courses have been shown to be great tools for the professional development of educators (Misra, 2018; Vivian et al., 2014), we have received a lot of feedback requesting additional support for more locally specific or introductory-level materials on TinyML that go beyond the open-source edX course materials on GitHub. Through our working groups we have been collaboratively developing ways to, for example, adapt our materials to fit into the Backyard Brains neuroscience curricula, and to teach courses in Portuguese (Rovai, 2021). Also, with the recent integration of CodeCraft and Edge impulse it is now possible to develop TinyML applications using visual programming abstractions which will open up new exciting opportunities to bring TinyML into the K12 classroom. Moving forward, we are excited to continue working to make TinyML accessible for all.

9. Limitations of Our Approach

We believe TinyML is an effective means to widen access to applied ML. Indeed, it is one way but not the only way. To provide a more balanced viewpoint, we describe some limitations of our approach and suggest alternative methods that may be more suitable.

Hardware cost. TinyML requires the purchase of embedded hardware to acquire the full-stack ML-development experience. The TinyML kit we developed costs US$49.99. In some developing

countries, this exceeds the average income in a week, in some rare cases, even a month. Although this price is considered reasonable in some countries, it may still be too high in others. We have found that the cost of shipping to distant parts of the world depends heavily on the presence of nearby distribution centers that carry the device. If none exist, the kit's cost, including shipping, can sometimes double the original kit price.

To help placate this issue we have, for example, arranged for sponsorship of 20 kits per University in our TinyML4D Academic Network and are aiming to pilot a kit sharing program between a handful of universities in the coming year. We are also experimenting with open-source emulation platforms such as Renode.io from Antmicro (Antmicro, 2021). Renode is an open-source framework that allows users to build and test embedded (ML) software without physical embedded hardware. Although this approach eliminates the hardware cost, students miss the opportunity and excitement of interacting with a device.

Course and Device accessibility. Globally, the number of embedded devices far exceeds the number of cloud and mobile devices (as Figure 4 shows). However, individuals must procure specific embedded hardware, such as the TinyML kit that we have developed with Arduino, to learn. By comparison, devices such as laptop and desktop computers connected to the web benefit from easier access. Students can use a regular computer to gain access to the online course materials. Even if they lack immediate access to computers in their homes, they can often access the online resources from Internet cafés that provide web access for a nominal fee. However, if we only leverage computers with web browsers, learners will have difficulty experiencing the complete ML workflow (Figure 5), since they will be unable to deploy in a device the models they train in the cloud.

Smartphones may be a suitable compromise. They are highly accessible, even though they can be an order of magnitude more costly than the TinyML kit. Nevertheless, they enable students to experience the complete TinyML design, development, deployment, and management workflow. Also, an average smartphone has more than 10 sensors—many more than the Arduino Nano 33 BLE Sense we use in Course 3, enabling additional applications. Learners can hold the smartphone in their hands, much like the TinyML device. That said, conveying the significance of ML's future being tiny and bright (Section 3) is more challenging (though not impossible) because mobile devices have far more resources (compute power, memory, bandwidth, etc.) than TinyML devices (Table 1). Students may therefore miss the fundamental issue of embedded-resource constraints. But if the goal is ultimately to expand access to applied ML, mobile devices may be a fair compromise.

Finally, it is important to note that these learners who have the hardest time accessing the course are exactly those who may benefit the most from the low-power and no-connectivity abilities of TinyML. As such we must find more ways to provide access in the future. The TinyML4D working group of TinyMLedu has begun to tackle this issue by creating a global network supported by UNESCO, whose goal is to "turn the pursuit of knowledge into an instrument for global peace and understanding." We hope that through this collaboration we can bring TinyML to everyone.

Programming background. Building ML models for mobile devices with TensorFlow Lite (Google, 2021) or the web using TensorFlow.js (Smilkov et al., 2019) is possible using high-level programming languages such as Python and JavaScript, respectively. These languages are easy to learn and far more accessible to beginners than C/C++, which is necessary to program embedded hardware (similar to Course 3). So although TinyML creates an opportunity to showcase the full-stack ML experience using embedded hardware, and we leverage the Arduino IDE and heavily scaffolded code with video walkthroughs to minimize the lift to C/C++, it may also narrow access in some regards. The additional necessary programming skills and associated education can be a roadblock.

In the future, we believe that end-to-end developer platforms such as Edge Impulse ("Edge Impulse", 2021; "Introduction to Embedded Machine Learning | Coursera", 2021) that lower the entry barrier into deploying ML on-device will become mainstream. Such platforms will become an essential part of the future developer ecosystem. Not every embedded ML engineer must know and understand all of the inner workings of TensorFlow Lite Micro or how an ML compiler works or how to extract the best performance from a highly customized ML hardware accelerator etc. Instead, learners need the right level of abstraction that allows them to focus on what matters most. Platforms such as Edge Impulse make it easy for learners, software developers, engineers and other domain experts to solve real-world problems using ML on the edge and TinyML devices without advanced degrees in ML or embedded systems. We therefore expose learners to the end-to-end MLOps platforms in Course 4, but note that more focus on such platforms in future courses could enable even more accessibility.

In summary, there are many paths to broaden applied-ML access. The correct approach–or, better, the most suitable approach–depends on the situation. We, therefore, hope this discussion clarifies the pros and cons of approaching applied ML through TinyML.

10. Related Work

There a wide variety of AI/ML-related courses that are currently being offered at all education levels. In this section, we compare and contrast how our approach differs from these works.

Online Content. There are several online ML courses offered via MOOC platforms such as Coursera and edX. Some of the most popular courses include "Machine Learning" by Andrew Ng on Coursera, "Machine Learning Crash Course" by Google AI and "Machine Learning" on edX ("Top 7 Machine Learning Courses - 2021 Guide and Reviews", 2021). We believe that these courses, along with many other excellent courses, differ from our approach across three primary categories: (1) theoretical vs. applied, (2) pedagogical vs. practical, and (3) conceptual vs. hands-on. No one approach is better than another. We believe that different offerings meet different needs.

Theoretical vs. Applied. The vast majority of ML courses focus on how the algorithms work mathematically. Understanding these underpinnings is crucial, but the extent to which one must master these fundamentals varies. For instance, a computer scientist who develops ML algorithms and writes computer programs in high-level languages like Python or C does not necessarily need to understand the details of the processor's microarchitecture or of the auto-differentiation engine used for backpropagation. However, he or she does need to know that such components exist and that there are domain experts who engineer such systems, providing API and ABI abstractions. This allows us to build systems at scale, leveraging the expertise of different communities. In a similar vein, we believe that not all ML courses must equally focus on the mathematical underpinnings.

The TinyML edX series focuses on the engineering principles needed to apply ML in practice. We preserve the mathematical underpinnings by providing learners with an intuitive and conceptual understanding of concepts such as stochastic gradient descent and the chain rule for backpropagation. We focus our energy on the end-to-end ML workflow (Figure 5), starting from dataset engineering to deploying models on embedded devices and everything in between. As Section 2 explains, there is a need for full-stack ML experience and our approach address this growing need.

Conceptual vs. Hands-on. ML courses often tend to focus on conceptual material with little to no hands-on deployment of models onto real devices. Usually, this is simply due to the lack of access to real-world deployment scenarios. For instance, one can rarely access a large-scale data center to deploy inference at scale in a cloud computing environment. Consequently, ML experience without hands-on deployment leads to knowledge gaps where learners understand how to build and

train ML models but cannot effectively reason about the challenges that arise when these models are deployed on a device in the field and experience the difficulties that emerge first-hand.

Thanks to the TinyML course kit (Section 4.7), it is easy for our learners to deploy their ML models on an actual embedded device that they can hold in the palm of their hand. Deployment allows learners to get exposure to when the models perform well and when they perform poorly. For instance, when learners train the person detection model in the VWW task and deploy it on the Arduino Nano 33 BLE Sense (Section 4), we ask them to evaluate how well the model performs under various lighting conditions. The lighting significantly affects the accuracy of the model because the pixels look different to the ML model. Similarly, when we ask our learners to train and deploy the keyword detection model in the KWS task (Section 4), we ask the learners to evaluate the device with their friends and family. Finally, Course 4 introduces learners to the diverse issues faced by applied ML engineers when they manage large-scale ML deployments.

However, this hands-on approach does come with some tradeoffs. First of all, it required a more narrow scope than some of the other related courses. For example, while we do mention the K-Nearest Neighbor Algorithm in our Anomaly Detection section of Course 2, for the most part, the courses focus on neural networks in the context of embedded systems. This narrower scope enables a deeper understanding of the full ML workflow and development cycle but prevents the course from covering a very broad set of ML techniques or cloud based deployments.

Pedagogical vs. Practical. On one hand, ML courses that are homebrewed at academic institutions often tend to focus on solid pedagogy, often diving deep into the details. But in doing so, it is easy to miss the forest for the trees. Most learners are not exposed to real-world issues. For example, rarely do ML classes teach the art of data engineering. Datasets are often pre-packaged and handed to learners, when in fact, the success of an ML model greatly depends on the dataset quality. On the other hand, industry certificate programs, such as Google's TensorFlow certificate program ("TensorFlow Developer Certificate", 2021), focus heavily on the practical aspects of training and deploying models. Both of these extremes have their pros and cons.

We take a more balanced approach. Being academic, we provide a solid pedagogical approach. By working closely with our industry counterparts, we bridge our pedagogical approach with practical applications of ML, as witnessed in the real world. We designed the TinyML course based on examples of real-world applications. KWS and VWW, for instance, are real-world use cases that many of us interact with daily. Because we value real-world use cases, we focus on data engineering principles and have exercises where our learners have to collect (and clean) their own datasets for keyword detection. Also, we blend software (algorithms) and hardware (embedded systems) with the product life cycle and ethical AI considerations as needed in practice.

Hands-on, lab-based MOOCs. It is well accepted that hands-on learning with lab components is an effective teaching method as it helps learners reinforce core concepts and build deeper understanding (Krajcik & Blumenfeld, 2006; Vesikivi et al., 2020). However, teaching hands-on learning with a lab component online, globally, is logistically challenging. As such, existing MOOCs that focus on hands-on embedded systems ("An Introduction to Programming the Internet of Things (IOT) Specialization", 2021; "Embedded Systems - Shape The World: Microcontroller Input/Output", 2021; "Embedded Systems - Shape The World: Multi-Threaded Interfacing", 2021; "Introduction to Embedded Systems Software and Development Environments", 2021; "IoT System Design: Software and Hardware Integration", 2021; "Professional Certificate in Embedded Systems Essentials with Arm", 2021) all rely on a low cost development kits from Texas Instruments, STMicroelectronics, Raspberry Pi, and Arduino. In fact, some of those courses also worked directly with

the suppliers to develop and efficiently distribute the kits. Building on this model, we worked with Arduino to develop the Tiny Machine Learning Kit (Section 4.7) and to provide global distribution.

Our courses differ from these other hands-on embedded systems courses because they are all core embedded systems courses, whereas our course is focused on applying ML on embedded devices. For example, "*Embedded Systems - Shape The World: Microcontroller Input/Output*" from The University of Texas at Austin ("Embedded Systems - Shape The World: Microcontroller Input/Output", 2021), covers a variety of core embedded systems topics such as building and testing circuits with switches, LEDs, resistors, potentiometers, and liquid crystal displays, as well as synchronizing hardware and software input and output, learning to solve problems using a finite state machine and debugging using oscilloscopes, logic analyzers, and software instrumentation. In contrast, the TinyML edX series strongly focuses on pedagogy at the intersection of ML, embedded systems and applications. To successfully deploy TinyML models, we touch on first-order embedded system principles, such as their resource constraints, how to program them and so forth. But the vast majority of our focus is on the challenges with running ML models on embedded devices, the embedded software frameworks for running ML on-device, and TinyML related optimizations.

K-12 At the K-12 level, there are many different active initiatives designed at bringing ML into the classroom. The primary focus of these efforts is on fast and easy ways to create ML models with little to no ML expertise or coding required. The goal is to introduce kids to the emerging world around them while inspiring them to pursue careers in STEM fields. AI4K12 ("AI4K12", 2021) is developing national guidelines for AI education for the K-12 education levels. The initiative is also curating resources to facilitate AI/ML instruction. They are also building a community of practitioners, researchers, resources and tool developers specifically focused on the AI for K-12 audience. Machine Learning for Kids ("Machine Learning for Kids", 2021) is a web-based interface that introduces ML to kids by providing hands-on experiences for training ML systems and building exciting projects with them. ML for kids builds on existing pedagogical efforts to introduce and teach programming to kids by adding ML models to educational coding platforms Scratch ("Scratch", 2021) and App Inventor ("MIT App Inventor", 2021) and helping children create projects and build games with the machine learning models they train. For example, kids learn to create a character in Scratch that smiles when the end-user types or says nice things to it and cries if mean things are said to it. In addition to these activities, numerous cross-curricular activities empower and support students as they progress through middle school, high school, and university using Arduino.

One of our primary goals is to introduce ML concepts and provide hands-on training to our learners. To this end, our learners are required to know some minimal level of programming, preferably in Python and C, albeit there are some limitations to our approach as presented in Section 9. The vast majority of our learners are beyond the K-12 level (Figure 14), although quite a few high-school students are taking the TinyML series. We are currently working on efforts through our TinyML4K12 and TinyML4STEM initiative to introduce the TinyML curriculum at a suitable level for younger learners. In the end, our goal is to inspire a new generation of learners that understand how to deploy technology responsibly to make the world a better place for everyone.

11. Conclusion

Expanding access to high-quality educational content, especially for machine learning, is important to ensuring that expertise diffuses beyond just a few prominent organizations. But doing so in a way that is both accessible and affordable to many different people is a difficult task. The four-part TinyML edX series we present here aims to tackle these challenges by providing application-driven content that covers the entire ML life cycle, giving students hands-on experience guided by world

experts and developing their ML skills regardless of their background. The forums, chats, optional project, and online discussions with the class creators promote community development and continued learning. The early impact of this approach is demonstrable: numerous participants from a variety of locations and demographics have signed up. We have also begun initiatives to further increase access by helping develop courses that target K–12 students and teachers, as well as courses in other languages.

Acknowledgements

Our approach to broadening access to applied machine learning using TinyML is based on input from many individuals at various organizations. We thank Rod Crawford, Tomas Edsö, and Felix Thomasmathibalan from **Arm**; Joe Lynas and Sacha Krstulović from **Audio Analytic**; Joshua Meyer from **Coqui**; Adam Benzio, Jenny Plunkett, Daniel Situnayake and Zach Shelby from **Edge Impulse**; Tulsee Doshi, Josh Gordon, Alex Gruenstein, Prateek Jain, and Nat Jeffries from **Google**; Marco Zennaro from **International Centre for Theoretical Physics (ICTP)**; Sek Chai from **Latent.AI**; Jane Polak Scowcroft from **Mozilla Common Voice**; Thiery Moreau from **OctoML**; Evgeni Gousev and Erich Plondke from **Qualcomm**; and Danilo Pau from **STMicroelectronics** for their valuable feedback. We are also grateful to the **Google TensorFlow Lite Micro team**, which includes Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, and Tiezhen Wang, without whom we would have been unable to deploy models in microcontrollers, and **Arduino**—Jose Garcia Dotel and Martino Facchin—who helped us with global distribution of the TinyML kit. We also thank the tinyML Foundation for nurturing activity around embedded ML, providing guidance and supporting educational and outreach activities around TinyML.

References

Ahmad, M. A., Eckert, C., & Teredesai, A. (2018). Interpretable machine learning in healthcare. *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics.*

AI4K12. (2021). https://ai4k12.org/

Alenezi, H. S., & Faisal, M. H. (2020). Utilizing crowdsourcing and machine learning in education: Literature review. *Education and Information Technologies.*

Algorithmia. (2020). Algorithmia 2020: State of enterprise ml.

An Introduction to Programming the Internet of Things (IOT) Specialization. (2021). https://www.coursera.org/specializations/iot

Antmicro. (2021). Renode.io - Antmicro's virtual development framework for complex embedded systems. https://renode.io/

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M., & Weber, G. (2020). Common Voice: A Massively-Multilingual Speech Corpus.

Arduino Nano 33 BLE [(Accessed on 04/02/2021)]. (2021). https://store.arduino.cc/usa/nano-33-ble

Arduino Tiny Machine Learning Kit. (2021). https://store.arduino.cc/usa/tiny-machine-learning-kit

Arribada Initiative: Open Source Conservation Technology. (2021). https://arribada.org/

Bahia, K., & Suardi, S. (2020). The state of mobile internet connectivity 2920. *GSMA Connected Society: London.*

Baldi, P. (2012). Autoencoders, unsupervised learning, and deep architectures. *Proceedings of ICML workshop on unsupervised and transfer learning.*

Belleflamme, P., & Jacqmin, J. (2014). An Economic Appraisal of MOOC Platforms: Business Models and Impacts on Higher Education. *CESifo Economic Studies*, *62*. https://doi.org/10.1093/cesifo/ifv016

Bisong, E. (2019). Google colaboratory. *Building machine learning and deep learning models on google cloud platform*. Springer.

Brigham, E. O. (1988). *The fast Fourier transform and its applications*. Prentice-Hall, Inc.

Brown, T. (2019). The AI Skills Shortage. Retrieved April 2, 2021, from https://itchronicles.com/artificial-intelligence/the-ai-skills-shortage/

Bruner, J. S. (2009). *The process of education*. Harvard University Press.

Buczak, A. L., & Guven, E. (2015). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, *18*(2).

Buolamwini, J., & Gebru, T. (2018). Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In S. A. Friedler & C. Wilson (Eds.), *Proceedings of the 1st conference on fairness, accountability and transparency*. PMLR. http://proceedings.mlr.press/v81/buolamwini18a.html

Canada, R. D. (2014). Audio Quality vs. Video Quality. *Rain Dance*. https://www.youtube.com/watch?v=-PLMiA18tBc

Capgemini Worldwide. (2020). The ai-powered enterprise.

Cervone, H. F. (2009). Applied digital library project management: Using Pugh matrix analysis in complex decision-making situations. *OCLC Systems & Services: International digital library perspectives*.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, *41*(3).

Chen, C., Seff, A., Kornhauser, A., & Xiao, J. (2015). Deepdriving: Learning affordance for direct perception in autonomous driving. *Proceedings of the IEEE international conference on computer vision*.

Chen, C.-M., & Wu, C.-H. (2015). Effects of different video lecture types on sustained attention, emotion, cognitive load, and learning performance. *Computers & Education*, *80*.

Choi, S., Seo, S., Shin, B., Byun, H., Kersner, M., Kim, B., Kim, D., & Ha, S. (2019). Temporal convolution for real-time keyword spotting on mobile devices. *arXiv preprint arXiv:1904.03814*.

Chowdhery, A., Warden, P., Shlens, J., Howard, A., & Rhodes, R. (2019). Visual wake words dataset. *arXiv preprint arXiv:1906.05721*.

Chuvpilo, G. (2020). Who's ahead in AI research AT NeurIPS 2020? https://chuvpilo.medium.com/whos-ahead-in-ai-research-at-neurips-2020-bf2a40a54325

Cloud automl custom machine learning models | google cloud [(Accessed on 05/30/2021)]. (2021). https://cloud.google.com/automl

David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., et al. (2020). Tensorflow lite micro: Embedded machine learning on tinyml systems. *arXiv preprint arXiv:2010.08678*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*.

Di Minin, E., Fink, C., Tenkanen, H., & Hiippala, T. (2018). Machine learning for tracking illegal wildlife trade on social media. *Nature ecology & evolution*, *2*(3).

Discourse Discussion Forum: How will the lack of female voices be handled. (2019). https://discourse.mozilla.org/t/how-will-the-lack-of-female-voices-be-handled/40551

Dogan, Ü., Edelbrunner, J., & Iossifidis, I. (2011). Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior. *2011 IEEE International Conference on Robotics and Biomimetics*.

Duhart, C., Dublon, G., Mayton, B., Davenport, G., & Paradiso, J. A. (2019). Deep learning for wildlife conservation and restoration efforts. *36th International Conference on Machine Learning, Long Beach*, *5*.

Dyrberg, N. R., & Holmegaard, H. T. (2019). Motivational patterns in STEM education: a self-determination perspective on first year courses. *Research in Science & Technological Education*, *37*(1).

Edge impulse [(Accessed on 06/02/2021)]. (2021). https://www.edgeimpulse.com/

Embedded Systems - Shape The World: Microcontroller Input/Output. (2021). https://www.edx.org/course/embedded-systems-shape-the-world-microcontroller-i

Embedded Systems - Shape The World: Multi-Threaded Interfacing. (2021). https://www.edx.org/course/embedded-systems-shape-the-world-multi-threaded-in

Frame.io. (2021). *Video Review and Collaboration Software*. https://www.frame.io/

Fraser, K. C., Meltzer, J., & Rudzicz, F. (2016). Linguistic Features Identify Alzheimer's Disease in Narrative Speech. *Journal of Alzheimer's disease : JAD*, *49 2*.

Gagné, J.-F. (2019). Global AI Talent Report 2019. Retrieved April 2, 2021, from https://jfgagne.ai/talent-2019/

The global chip shortage: A timeline of unfortunate events [(Accessed on 12/02/2021)]. (n.d.).

Google. (2020a). Responsible AI Toolkit TensorFlow. https://www.tensorflow.org/responsible%5C%5Fai

Google. (2020b). What-If Tool. https://pair-code.github.io/what-if-tool/

Google. (2021). Tensorflow lite | ml for mobile and edge devices [(Accessed on 06/02/2021)]. https://www.tensorflow.org/lite

Gopalakrishna, A. K., Özçelebi, T., Liotta, A., & Lukkien, J. J. (2012). Exploiting machine learning for intelligent room lighting applications. *2012 6th IEEE International Conference Intelligent Systems*.

Grosz, B. J., Grant, D. G., Vredenburgh, K., Behrends, J., Hu, L., Simmons, A., & Waldo, J. (2019). Embedded ethiCS: Integrating ethics across CS curriculum [https://cacm.acm.org/magazines/2019/8/238345-embedded-ethics/fulltext]. *Communications of the ACM*, *62*(8).

Gruenstein, A., Alvarez, R., Thornton, C., & Ghodrat, M. (2017). A cascade architecture for keyword spotting on mobile devices. *arXiv preprint arXiv:1712.03603*.

Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003). KNN model-based approach in classification. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*.

Guo, P. J., Kim, J., & Rubin, R. (2014). How video production affects student engagement: An empirical study of MOOC videos. *Proceedings of the first ACM conference on Learning scale conference*.

Gupta, S., Kazi, F., Wagh, S., & Kambli, R. (2015). Neural network based early warning system for an emerging blackout in smart grid power networks. *Intelligent distributed computing*. Springer.

Harden, R. M. (1999). What is a spiral curriculum? *Medical teacher*, *21*(2).

Harwell, D. (2019). *Doorbell-camera firm Ring has partnered with 400 police forces, extending surveillance concerns*. https://www.washingtonpost.com/technology/2019/08/28/doorbell-camera-firm-ring-has-partnered-with-police-forces-extending-surveillance-reach/

Hollands, F., & Kazi, A. (2019). Benefits and Costs of MOOC-Based Alternative Credentials: 2018-2019 Results from End-of-Program Surveys. *Center for Benefit-Cost Studies of Education, Teachers College, Columbia University*.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *CoRR*, *abs/1704.04861*. http://arxiv.org/abs/1704.04861

IC Insights. (2020). *MCUs Expected to Make Modest Comeback after 2020 Drop*. https://www.icinsights.com/news/bulletins/MCUs-Expected-To-Make-Modest-Comeback-After-2020-Drop--/

Introduction to embedded machine learning | coursera [(Accessed on 06/09/2021)]. (2021). https://www.coursera.org/learn/introduction-to-embedded-machine-learning

Introduction to Embedded Systems Software and Development Environments. (2021). https://www.coursera.org/learn/introduction-embedded-systems

IoT System Design: Software and Hardware Integration. (2021). https://www.edx.org/course/iot-system-design-software-and-hardware-integratio

Jahangiri, A., & Rakha, H. A. (2015). Applying machine learning techniques to transportation mode recognition using mobile phone sensor data. *IEEE transactions on intelligent transportation systems*, *16*(5).

Jamieson, P. (2011). Arduino for teaching embedded systems. are computer scientists and engineering educators missing the boat? *Proceedings of the international conference on frontiers in education: computer science and computer engineering (FECS)*.

Johnson, J. (2021). Global digital population as of January 2021. https://www.statista.com/statistics/617136/digital-population-worldwide/%5C#:~:text=Global%5C%20internet%5C%20usage%5C&text=The%5C%20global%5C%20internet%5C%20penetration%5C%20rate,penetration%5C%20rate%5C%20among%5C%20the%5C%20population.

Koenecke, A., Nam, A., Lake, E., Nudell, J., Quartey, M., Mengesha, Z., Toups, C., Rickford, J. R., Jurafsky, D., & Goel, S. (2020). Racial disparities in automated speech recognition. *Proceedings of the National Academy of Sciences*, *117*(14). https://doi.org/10.1073/pnas.1915768117

Kolachalama, V. B., & Garg, P. S. (2018). Machine learning and medical education. *NPJ digital medicine*, *1*(1).

Krajcik, J. S., & Blumenfeld, P. C. (2006). *Project-based learning*. na.

Lockman, A. S., & Schirmer, B. R. (2020). Online Instruction in Higher Education: Promising, Research-Based, and Evidence-Based Practices. *Journal of Education and E-Learning Research*, *7*(2).

Machine Learning for Kids. (2021). https://machinelearningforkids.co.uk/

Maedche, A., Legner, C., Benlian, A., Berger, B., Gimpel, H., Hess, T., Hinz, O., Morana, S., & Söllner, M. (2019). AI-based digital assistants. *Business & Information Systems Engineering*, *61*(4).

Martin, T. (2016). *The designer's guide to the Cortex-M processor family*. Newnes.

Mayer, R., & Mayer, R. E. (2005). *The Cambridge handbook of multimedia learning*. Cambridge university press.

Mayer, R. E., Fiorella, L., & Stull, A. (2020). Five ways to increase the effectiveness of instructional video. *Educational Technology Research and Development*, *68*(3).

Mbed OS [(Accessed on 04/02/2021)]. (2021). https://os.mbed.com/mbed-os/

Misra, P. (2018). Moocs for teacher professional development: Reflections and suggested actions. *Open Praxis*, *10*(1), 67–77.

MIT App Inventor. (2021). https://appinventor.mit.edu/

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model Cards for Model Reporting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*. https://doi.org/10.1145/3287560.3287596

Mitchell, T. M., Caruana, R., Freitag, D., McDermott, J., Zabowski, D., et al. (1994). Experience with a learning personal assistant. *Communications of the ACM*, *37*(7).

Neumann, Y., Neumann, E., & Lewis, S. (2017). The robust learning model with a spiral curriculum: Implications for the educational effectiveness of online master degree programs. *Contemporary Issues in Education Research*, *10*(2).

nRF52840 [(Accessed on 04/02/2021)]. (2021). https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52840-SoC-product-brief.pdf?la=en%5C&hash=EDF4C48A053E7943AD3C9DD3963B626D768B5885

OV7675 [(Accessed on 04/02/2021)]. (2021). https://www.arducam.com/products/camera-breakout-board/0-3mp-ov7675/

Pashler, H., McDaniel, M., Rohrer, D., & Bjork, R. (2008). Learning styles: Concepts and evidence. *Psychological science in the public interest*, *9*(3).

Pico4ML by Arducam [(Accessed on 04/02/2021)]. (2021). https://www.arducam.com/pico4ml-an-rp2040-based-platform-for-tiny-machine-learning/

Plancher, B., & Janapa Reddi, V. (2022). Tinymledu: The tiny machine learning open education initiative. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*. https://doi.org/10.1145/3478432.3499093

Prabhu, K., Jun, B., Hu, P., Asgar, Z., Katti, S., & Warden, P. (2021). Privacy-Preserving Inference on the Edge: Mitigating a New Threat Model. *Research Symposium on Tiny Machine Learning*. https://openreview.net/forum?id=rlHeH9tx3SM

Professional Certificate in Embedded Systems Essentials with Arm. (2021). https://www.edx.org/professional-certificate/armeducationx-embedded-systems-essentials

Reinheimer, B., Aldag, L., Mayer, P., Mossano, M., Duezguen, R., Lofthouse, B., von Landesberger, T., & Volkamer, M. (2020). An investigation of phishing awareness and education over time: When and how to best remind users. *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*.

Rovai, M. (2021). TinyML - Machine Learning for Embedding Devices. https://github.com/Mjrovai/UNIFEI-IESTI01-T01-2021.1

Schubert, T. W., D'Ausilio, A., & Canto, R. (2013). Using Arduino microcontroller boards to measure response latencies. *Behavior research methods*, *45*(4).

Scratch. (2021). https://scratch.mit.edu/

Shor, J., Emanuel, D., Lang, O., Tuval, O., Brenner, M., Cattiau, J., Vieira, F., McNally, M., Charbonneau, T., Nollstadt, M., Hassidim, A., & Matias, Y. (2019). Personalizing ASR for Dysarthric and Accented Speech with Limited Data.

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S. N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F. B., & Wattenberg, M. (2019). Tensorflow.js: Machine learning for the web and beyond. *arXiv preprint arXiv:1901.05350*.

Smith, A. (2018). *Public Attitudes Toward Technology Companies*. https://www.pewresearch.org/internet/2018/06/28/public-attitudes-toward-technology-companies/

Stack Overflow Developer Survey. (2020). https://insights.stackoverflow.com/survey/2020%5C#developer-roles

Systems, E. (2021). ESP32. https://www.espressif.com/en/products/socs/esp32

TensorFlow. (2021). TinyConv [[Online; accessed 15-Mar-2021]]. https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/speech%5C_commands/models.py

TensorFlow Developer Certificate. (2021). https://www.tensorflow.org/certificate

Tiwari, V. (2010). MFCC and its applications in speaker recognition. *International journal on emerging technologies*, *1*(1).

Top 7 Machine Learning Courses - 2021 Guide and Reviews. (2021). https://www.learndatasci.com/best-machine-learning-courses/

Turnbull, A., Carroll, J., & McDonald, A. (2021). Combining SCADA and vibration data into a single anomaly detection model to predict wind turbine component failure. *Wind Energy*, *24*(3).

Turner, A.-M. (2021). 10 free online classes from Harvard to learn something new. https://mashable.com/article/free-harvard-classes-online/

Vesikivi, P., Lakkala, M., Holvikivi, J., & Muukkonen, H. (2020). The impact of project-based learning curriculum on first-year retention, study experiences, and knowledge work competence. *Research Papers in Education*, *35*(1).

Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: Moocs as a scalable solution for teacher professional development.

Wang, J., Antonenko, P., & Dawson, K. (2020). Does visual attention to the instructor in online video affect learning and learner perceptions? An eye-tracking analysis. *Computers & Education*, *146*.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Whittaker, Z. (2018). *Fitness app PumpUp leaked health data, private messages*. https://www.zdnet.com/article/fitness-app-pumpup-leaked-health-data-private-messages/

Wladis, C., Hachey, A. C., & Conway, K. (2014). An investigation of course-level factors as predictors of online STEM course outcomes. *Computers & Education*, *77*.

Wu, Y., Rough, D., Bleakley, A., Edwards, J., Cooney, O., Doyle, P. R., Clark, L., & Cowan, B. R. (2020). See What I'm Saying? Comparing Intelligent Personal Assistant Use for Native and Non-Native Language Speakers. *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. https://doi.org/10.1145/3379503.3403563

Yairi, T., Kawahara, Y., Fujimaki, R., Sato, Y., & Machida, K. (2006). Telemetry-mining: a machine learning approach to anomaly detection and fault diagnosis for space systems. *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*.

Yang, D. (2017). Instructional strategies and course design for teaching statistics online: perspectives from online students. *International Journal of STEM Education*, *4*(1).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Zantalis, F., Koulouras, G., Karabetsos, S., & Kandris, D. (2019). A review of machine learning and IoT in smart transportation. *Future Internet*, *11*(4).

Zeng, A., Song, S., Lee, J., Rodriguez, A., & Funkhouser, T. A. (2019). TossingBot: Learning to Throw Arbitrary Objects with Residual Physics. *CoRR*, *abs/1903.11239*. http://arxiv.org/abs/1903.11239

Zeng, X., Yang, M., & Bo, Y. (2020). Gearbox oil temperature anomaly detection for wind turbine based on sparse Bayesian probability estimation. *International Journal of Electrical Power & Energy Systems*, *123*.

Zhang, G., Yan, C., Ji, X., Zhang, T., Zhang, T., & Xu, W. (2017). Dolphinattack: Inaudible voice commands. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*.

Zhang, Y., Suda, N., Lai, L., & Chandra, V. (2017). Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*.

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.