Teaching Neural Networks in Ten Weeks: Creating an Activity-Based and Accessible Course on Machine Learning for First-Year Undergraduates

Kevin William Bachelor, University California, Santa Cruz

Kevin Bachelor is a fourth year Computer Science student at University of California, Santa Cruz. Kevin taught Machine Learning as a student instructor for 2 years with the Lead by Design Program under Professor Tela Favaloro after finding his love for teaching by leading and teaching with the University's AI club. His teaching propensity for teaching ML comes from his fascination with the topic, and a background in acting. He continues each individually as well with a theater minor, and continued research in Machine Learning for Molecular Dynamics with Professor Razvan Marinescu. He will soon be continuing on as a PhD at UCSC under the same lab doing development in the intersection between machine learning and computational chemistry for drug design.

Anthony Furman, University of California, Santa Cruz

Anthony Furman is a Robotics Engineering undergraduate student and research assistant at the University of California, Santa Cruz. He joined the Lead by Design program Fall 2024 and designed the Introduction to Machine Learning class, then teaching it for two years. He has applied his interest in both Machine Learning and Robotics at the UCSC HARE Lab, where he works on designing RL algorithms for robotic systems.

Dr. Tela Favaloro, University of California, Santa Cruz

Tela Favaloro is an associate teaching professor for the Baskin School of Engineering at UCSC where she works to establish holistic interdisciplinary programming centered in experiential learning. Her Ph.D is in Electrical Engineering with emphasis in the design and fabrication of laboratory apparatus and techniques for electro-thermal characterization of sustainable power systems as well as the design of learner-centered experiential curriculum. She is currently working to develop an inclusion-centered first-year engineering program in hands on design and problem-based learning to better support students as they enter the engineering fields.

©American Society for Engineering Education, 2026

Teaching Neural Networks in Ten Weeks: Creating an Activity-Based and Accessible Course on Machine Learning for First-Year Undergraduates

Abstract

In this complete evidence-based practice paper, we detail the full workings of our class, *Introduction to Machine Learning*, a lower-division undergraduate course. Over the 10-week quarter, learners create several types of Neural Networks, culminating with their own customized architecture for an application of their choice, all without requiring any enrollment prerequisites. By focusing on a big-picture understanding of Machine Learning, we create an accessible and engaging learning environment that fills a gap in existing curricula and results in students achieving learning outcomes desired in industry. All of this is achieved within a 3-unit introductory course, despite that at the start of the class, only 18.8% of learners on average reported proficiency in Python and only 4.1% had worked with ML before. By the end of the class, students demonstrate strong ML skills, including processing datasets, creating custom models, effectively using remote server computation, solving ML problems, and applying statistical and mathematical understanding by predicting outcomes of changes to the model structure – all skills required of an ML professional.

Background

Our Introduction to Machine Learning Course seeks to fill a skill gap by teaching first year undergraduate students to create and deploy advanced neural networks in ten weeks. We do so by focusing on holistic understanding of Machine Learning (ML) concepts without math prerequisites, resulting in students producing several ML models from scratch. The 10-week, 3-unit course culminates in a multi-week final, wherein students create a Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN) in PyTorch using a dataset they find independently. The course is specifically built to maximize accessibility by having no prerequisites, while also teaching both theoretical ML concepts such as gradient descent, neural networks, convolutions, as well as applied techniques in model training, data sanitization and analysis, and remote server computation. In this paper, we will discuss the benefits, methods, and results of our ML course and the unique teaching paradigm it effectively utilizes.

While much of today's groundbreaking technology incorporates what is often labeled "Artificial Intelligence (AI)," this term is often vaguely used to refer to a variety of different technologies. Meanwhile, the term "Machine Learning" specifically describes the computer science paradigm in which a computer program learns to provide useful outputs through training. These outputs could be as simple as labeling what is in an image or more complex, such as outputting the correct response to a user's query. While usually classified as a subset of AI, ML has grown large enough that the terms are often used interchangeably. However, as the proper terminology for the technology driving advances such as LLMs, Gen AI, Computer Vision, recommendation algorithms, etc. is specifically Machine Learning [1], we will use this term for the remainder of the paper.

Over the past decade, there has been a sharp increase in demand for ML skills in industry and

research. The surge in demand has resulted in workforce positions remaining unfilled, specifically Reuters estimated a hiring gap of 50% and a skill deficiency among 70% of workers in many ML-related skills for 2024 [2]. A 2026 International Monetary Fund report of several hundred million job postings between 2010 and 2025 also highlights the demand for ML. The report found that key ML skills related to "building and deploying [ML] models" went from approximately 0.01% of online job postings to over 2% in the US between 2010 and 2015. Furthermore, the specified ML skills below (excluding possibly Data Analysis) were estimated to be four times as effective at increasing wages over learning any other skill in the US between 2020 and 2024 [3]. Notably, the report specifically excludes skills related to LLM usage for these metrics. Instead, it directly lists the following examples of key in-demand ML skills:

- Building/Developing ML models
- Deploying ML Models (ML-Ops)
- Data Analysis
- Python for Machine Learning, PyTorch
- Evaluating ML Models

This influx in demand is further evidenced by the surge in salaries for ML-based jobs compared to other non-ML-based jobs in the computer science field. In particular, ML Engineer salaries are 32-55% higher than that of the median software developers, such as front and backend developers [4–6]. Meanwhile in research, the Nobel Prizes in Physics and Chemistry, possibly the most prestigious award in their respective fields, were recently awarded to three machine learning contributions of Geoffrey Hinton [7], John Hopfield [8], and David Baker [9]; providing ample evidence of the impact of ML on today's world.

ML has become integrated with the world around us; not only is a highly skilled workforce in demand, but there should also be a more educated general populace. The use of ML technologies such as LLMs, automatic facial recognition, and smart recommendation systems have become ever present in an everyday life, regardless of a person's employment. Despite this, ML knowledge and skills are typically concentrated in only a few undergraduate majors within US-based universities [3]. As teachers, we have a responsibility to provide opportunities for everyone, not just those who are planning to become experts in the field, to learn the basics and have confidence in understanding the capabilities and inner workings of ML. Equipping people with an understanding of the world around them, regardless of their employment, is a key goal of education.

Training and education in ML is expected to be provided by the university system. Yet, existing coursework is typically gated behind multiple prerequisites, preventing students from accessing these in-demand skills until their final years of college. By not offering exposure to ML until only a year before graduation, universities hold back students from gaining a proper ML education, effectively removing the opportunities to use and thus refine their knowledge and skills. The majority of university-level classes that teach ML require a programming background, multivariate calculus, linear algebra, and probability courses; those that do not list all of these as a prerequisite are rare. Courses that have no prerequisites are virtually non-existent. This is evidenced by an analysis of 10 universities ranked in a list of "Top AI schools" [10], where not one offered an ML course with no prerequisites, despite each school having at least one or even several "introductory" level ML classes. In fact, of approximately 30 classes surveyed, none of

them offered a course with only programming as a prerequisite; all required some additional advanced math prerequisite course. From our experience, completing these prerequisite courses is necessary to understand the deepest levels of the mathematical workings of ML, but not to begin to work with ML models or understand their operations. As a result, a potential student's start in ML is held back by these prerequisites until their final years of college. Instead of ML being an integrated part of the curriculum, ML coursework is limited to an advanced elective. First, this makes the advanced ML courses more difficult for those interested in them, as they will have little theoretical foundation to build from. Furthermore, the long wait dissuades students who are unsure about going into the field as well as students from non-STEM majors from being involved in ML. The current course offerings effectively gatekeep understanding and exposure to ML from the general student population, failing both to provide a general education to all students, or a specific education to those interested in ML.

As an alternative to university education, many students turn to online resources to build their knowledge and skills. Unfortunately, these online resources are often disorganized and fail to offer a rigid structure needed to support most students' learning. Moreover, the sheer number of resources available poses a problem, as it overwhelms prospective students with choice, and makes it difficult to figure out which ones will be at a helpful level of difficulty and a high enough caliber. These materials are also rarely certified, meaning that if a student completes them, there is no guarantee that they learned a complete set of skills, thus, employers looking for education in ML may not trust online course completion. Even if a student does manage to stumble onto a useful online resource, the fact that it is online and typically asynchronous means that students have little scaffolding or external push to see it to the end. Most students who start an online resource find themselves without either the motivation or the time to complete it, limiting those who can effectively utilize them, as shown by their extremely low completion rates [11, 12].

Considering their limited options, students are also looking to extracurricular sources for Machine Learning education. Many R1 universities are now finding themselves with student-led Machine Learning clubs, a few of which are even specifically aimed at teaching AI to other students (i.e. UCSC's Santa Cruz AI Club, UC Berkeley's Machine Learning at Berkeley, Rice University's Machine Learning @ Rice, and University of Michigan's Michigan Data Science Team (MDST), among others). While helpful as a resource for students to fill a gap in their education, clubs have substantially fewer resources than a class, and therefore are limited in their breadth, depth, and accessibility of content. The popularity of these clubs is demonstrative evidence of the existence of a gap in university education and therefore a warning sign that something needs to be done to bring these skills more readily into the curriculum earlier.

As undergraduate students ourselves, we have a unique understanding of and access to students' opinions and views of the curricular track. Our own identities as students, club leaders, ML researchers, and student-teachers afford us insight not only into the many issues and struggles we've faced, but also those faced by our fellow students. This personal experience drove us to reconsider and redesign how ML might be taught. Calling upon our own conceptualizations and experience teaching, we actualized our ideas into *Introduction to Machine Learning*, a lower-division undergraduate, credit-bearing class at our university without prerequisites, supported by faculty within our School of Engineering [13–17].

In this paper, we offer a different approach to Machine Learning education, one that is accessible to any student. *Intro to ML* is designed to support undergraduate learners at an R1 university, especially those in their first year. It seeks to address these accessibility issues by flipping the learning pathway, starting with intuition- and context-building first before going into the theoretical workings and then the application of skills, abstracting heavy math topics. Our high-structure course achieves its goal of teaching competency in ML within a 10-week quarter without prerequisites through several experiential techniques.

Course Design

Clearly, a gap exists in current university methods for teaching ML. Our approach aims to solve this by flipping the typical pathway for ML education. Rather than first requiring strong math prerequisites, which are then used to describe ML models, *Intro to ML* first describes ML models, allowing students to gain an intuitive understanding of how they work to be later reinforced by rigorous mathematical definitions and proofs. This approach creates a more accessible curriculum so that all students can engage with the material and gain a holistic understanding of ML much earlier in their university career. By the end of our 10-week course, students are able to: make conclusions about what kind of problems are appropriate for ML, predict how models will react to certain changes in their structure, and program their own models for deployment in a self-selected context, as they do for the final project of this course. *Intro to ML* empowers students to be able to make meaningful decisions as to whether they want to continue in the field and how they can apply this ML knowledge to other fields of interest or daily life.

Designing a high-structure, experiential classroom makes this intuition-first and math-second approach possible. Experiential Learning is a well-tested methodology where the teacher facilitates students' learning through activities, experiences, and reflection, which has been shown to improve student outcomes [18–22]. Activities during class help break down barriers as students interact with concepts in a supportive, low-risk environment, building their own understanding alongside their peers. To further accessibility to the content, *Intro to ML* utilizes a high-structure course design. High-structure has been shown to close achievement gaps among learners by making content more accessible to all [20, 21] and thus, it drives our decision-making when designing the course. Each week of the class is broken down into discrete deadlines that serve to prepare learners, invite practice, and synthesize the learning outcomes of the week. As previously mentioned, ML is often gatekept and portrayed as only accessible to those with complex mathematical knowledge; high-structure and experiential learning allow us to overcome these barriers with a holistic, intuition-first approach.

Thanks to the uniquely effective "set them up and let them go" [13, 14] instructional design that high-structure and experiential learning create when used in tandem, we achieve a high quantity of non-trivial learning outcomes in one quarter, without prerequisites. *Intro to ML* supports learners in reaching advanced learning outcomes desired in the ML field by incrementally building knowledge with a regular and reliable pace, as shown in Table 1. Specifically, out-of-class preparatory material primes students to engage with in-class participatory learning meant to build intuition through concrete experience. Learners then independently synthesize learning through a design-based lab alongside a documentation tool that encourages critical thinking. In this way,

learners are set up to apply their learning during the midterm and final projects. We present our weekly structure graphically in Figure 1 and provide elaboration in the following section.

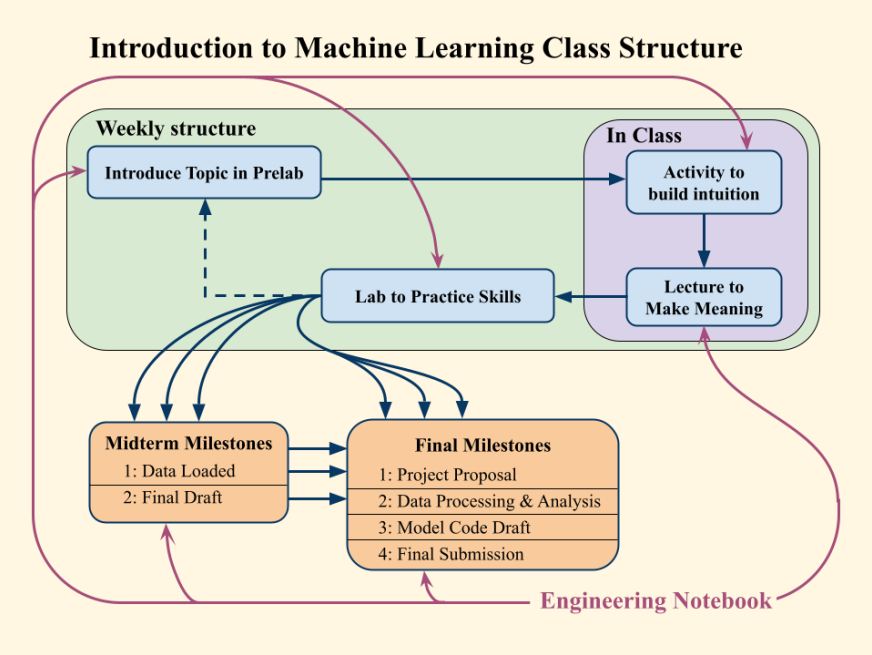

Figure 1: A diagram of the structure of the instructional design in Introduction to Machine Learning. Note that the weekly structure flows toward increased complexity of topics (see Table 1) while the notebook is used to support all content delivered.

Key Instructional Components

Prelabs: To build a common foundation of knowledge that allows for participation in the classroom, we assign pre-class materials, often a short reading or small task, before each class period. Prelabs not only serve to introduce topics but also to refresh students on anything necessary for the class [13, 14]. As many of the class concepts rely heavily on these materials having been completed, we check for understanding (and completion) by issuing a short open-note quiz at the start of class. Quizzes are completed on paper to avoid cheating with only the engineering notebook available as support, reinforcing the habit of notetaking.

Class time: After completing a prelab review, most of class time is used to facilitate intuition-building activities that elucidate the inner workings of key ML concepts, building on the theoretical foundation established in the prelabs. We address how these activities are run, their learning outcomes, and many of their benefits in the "Experiential Learning: Activity Design for the ML Classroom" section of this paper. Class time before activities is used to frame the incoming activity by introducing the task and its context. Lectures after an activity bridge intuition into concrete understanding through meaning-making, where we attach specific vocabulary to the concepts taught and reflect on key takeaways.

Labs: After collaboratively exploring the concepts in the classroom, the students practice and solidify what they learned in weekly labs, wherein we provide practice applications for them to

solve experientially. Labs are designed to ask reflection-based questions alongside content-based prompts; these tend to involve learners writing code and/or doing a small problem to improve the skills and knowledge they learned that week in the lessons, with a slightly different application to encourage critical thinking and analysis of how each part of the skill relates to each part of the problem. For example, one question in the Week 2 lab asks students to code a 3-input perceptron; the Week 5 lab asks them to draw a confusion matrix; students are tasked with generalizing the output of a 1-dimensional RNN to multiple dimensions in Week 8. Labs are accompanied by planning and reflection questions that ask for deeper thinking about their decisions and process.

Iterative Projects: The midterm (Week 6) and final (Weeks 7 - 10) both serve as opportunities to practice systems thinking and apply learning to demonstrate their current level of knowledge and skills. These projects in turn are each broken into regular milestones with documentation prompts to support steady and incremental progress while providing opportunities for formative assessment. The midterm project synthesizes the first five weeks of content by asking students to build a neural network independently: a housing price predictor with a dataset that we give them. Thus, students are given the problem but must come up with their own model architecture to solve it and then justify and reflect on their design decisions when creating the model. The final project takes their learning a step further by granting student-teams ownership over the goal for their ML model; they select a problem to solve through ML and the appropriate dataset and model architecture. Specifically, we ask them to design, construct, train, and present a CNN or RNN on a dataset they find and process. Their first milestone is to submit a detailed project proposal; each subsequent week has a deadline for a new goal, helping time box student progress and ensure consistent and continuous work (following our high-structure pedagogical approach).

Professional Documentation: We conduct continuous formative assessment, surface student thinking, and encourage thoughtful practice throughout the class through engineering notebook submissions wherein students document their process, as depicted in Figure 1. This documentation process is rigorously established through an introductory experiential activity and reinforced through regular habit; the labs always ask students to write a plan to tackle the assignment, record their process and challenges, and follow up their work with reflection, all in their notebooks. By creating documentation standards, we are able to see into student's thought processes and can then tailor the class on the fly to correct any confusion or misunderstandings. Meanwhile, students benefit from reflection and critical thinking practices throughout the class, which has been shown to support learning [21]. Additionally, as students begin to work on their final group projects, note-taking allows the teachers to assess individual student contributions to the team project, and step in when disparities in work arise.

Table 1: A comparison of our course's weekly learning outcomes to some of the most in-demand and domain specific skills according to 2026 IMF data [3] and supported by Reuters findings for 2024 [2]. Intro to ML teaches these in-demand skills without prerequisites through high-structure and experiential learning that builds in complexity over 10 weeks.

In Demand Skills for Machine Learning [2, 3]:
A: Understanding ML Models
B: Deploying models (ML Ops)
C: Data Analysis
D: Python for Machine Learning, PyTorch
E: Evaluating ML Models

Week #	***Intro to ML* Learning Outcomes, organized by week**	**A**	**B**	**C**	**D**	**E**
1	• Justify Machine Learning process Pipeline • Explain importance of mathematical models • Describe the Perceptron model	✓				
2	• Define the basic Neural Network Architecture • Analyze the importance of Activation Functions • Illustrate the process of Gradient Descent	✓				✓
3	• Employ various Python Data processing tools • Process and clean a dataset • Analyze data numerically and graphically			✓	✓	✓
4	• Design and construct a Neural Network in PyTorch • Implement the train-test-use pipeline in PyTorch • Justify basic Neural Network architecture decisions				✓	✓
5	• Create a dataset and data loader in PyTorch • Set up and utilize a validation step in PyTorch • Assess over/underfitting from validation			✓	✓	✓
6 Midterm	• Describe how an RNN processes sequential data • Identify the limiting architectural issue with RNNs • Express how a convolution works & its benefits	✓			✓	
7 Final	• Analyze a dataset for training viability • Practice framing and breaking down ML problems • Perform collaborative coding through git			✓	✓	✓
8 Final	• Describe how an LSTM fixes problems with RNNs • Convert arbitrary block diagrams into PyTorch code • Process images into data for training a model	✓			✓	
9 Final	• Design and Construct a CNN in PyTorch • Deploy a model on cloud infrastructure using MLOps • Recognize how a GPU speeds up machine learning	✓	✓		✓	
10 Final	• Fully construct a functional CNN based on custom data • Understand and employ dropout or normalization	✓	✓	✓	✓	✓

A primary example of *Intro to ML's* structure in practice is Week 2 content, where we show how the distributive property leads to functions (models) "collapsing" when composed unless activation functions are used. Before the lesson, the students watch a prelab video on the distributive property and experiment with function composition in the graphing tool Desmos, which will be used during class. To open class, students take a small 5-question quiz that asks them to apply the distributive property and describe function composition. We then run two activities during class. In the first activity, students gather around a whiteboard and help the instructor mathematically demonstrate that when constants are distributed, as in the prelab, they can "collapse" into simpler expressions. We then demonstrate how two layers of a neural network, as expressed with function composition, will similarly "collapse" into one when the distributive property is applied. During the second activity, students discover how the collapse can be avoided. With a Desmos simulation, students are given time to play with composing activation functions to provide barriers to model collapse. Immediately following the activities, we give a short lecture that formally defines activation functions and clarifies their importance. That week, the lab assignment tasks learners with describing where missing activation functions should go within a model. Concurrently, students document their learning in their engineering notebooks; we ask students to first describe and plan out how they will approach the coding section of the lab and then afterwards reflect on the sections they found most challenging. Later in the course, students must again use activation functions in the right place for their midterm, as well as their final project, drawing upon the knowledge gained in these lessons. When we reviewed their submissions for this quarter, we saw that some students struggled to understand that activation functions are needed after every hidden layer in neural networks with multiple layers. As a result, we planned a short review section at the beginning of the next class covering how to place activation functions in a neural network with many layers and the reasoning for their placement. In this way, high structure and a participatory environment enabled us to teach a large number of high demand concepts efficiently and accessibly.

Experiential Learning: Activity Design for the ML Classroom

In-class activities are critical in supporting early and rapid holistic understanding, as they are designed to put students in the perspective of the model itself. We follow several design philosophies and goals when formulating our activities. In this paper, we will focus on the three most influential ideas: 1) computer-free participation, 2) utilization of our mental models for concepts, and 3) enjoyment of learning.

We intentionally design many of our activities to be computer-independent. The removal of the computer allows students to become more immersed in the activity, utilizing existing "obvious" implicit knowledge of how physical reality works to learn ML while also encouraging collaboration. Student immersion is facilitated by physical participation, often with tangible objects. Even simply asking students to stand up and gather around the whiteboard facilitates engagement and collaboration. Simply put, physical activities encourage students to talk to each other to develop solutions and collaboratively construct knowledge. Beyond promoting collaboration, removing the computer leverages the students' preexisting complex understanding of how the real-world works, unlike computer applications which have an innate skill curve to learn, regardless of simplicity. By utilizing students' existing knowledge and avoiding the black

box that is often attributed to computer tools, students are almost forced to actively interact with the concepts and their peers.

Our *Becoming an ML Model* in-class activity, which is used to illustrate the *ML model pipeline* of training, testing, and being put into public use (deployment) in early weeks of the course, emphasizes the benefits of computer-independent pedagogy. For the activity, the class itself becomes the ML model performing the processes of train, test, and use on a 'dataset' of image-number pairs; the class is first shown the image and asked to guess the associated number. Then, the students are given the correct number for that image without any explanation as to the correlation, and the process continues for the next image-number. The hidden nature of the underlying connection between the image and its number places students into the perspective of an ML model during training, which will receive inputs and expected outputs without knowing how they are correlated. After guessing numbers for many images, the students eventually figure out that the number is related to the worth of the object in the image, just as an ML model eventually correlates the inputs and outputs in the data in the training phase. We then move on to the testing phase, where students try to guess the numbers for a series of new images that serve as a testing set, and their accuracy is scored and logged. Finally, they guess numbers for images that do not have corresponding numbers to represent what the use of an ML model in the real world is like. Each of these steps had students acting as the model would act in the train, test, or use stages of ML by limiting understanding of the data. Rather than being instructed that a model trains from input-output pairs, the learners went through the ML pipeline themselves. The students' experience of training on input-output pairs, testing to determine their accuracy as a model, and finally performing the use phase of making predictions on unlabeled data intuitively taught them the ML model pipeline by illustrating why each step occurs.

Another framework we utilize for designing activities comes from analyzing how Machine Learning concepts are structured and formulated within our minds, rather than how those concepts might be typically taught. Here, we bring our unique perspective as student-teachers; we specifically tailor outcomes towards building *mental models* that may be leveraged by students to more easily access higher-order concepts. Mental models provide a more intuitive and useful paradigm of thinking for students, while also being something that we can more effectively teach. Using our experiences as students ourselves, we worked to bring the way in which knowledge is constructed in our own minds into the physical world. We were therefore able to teach complex concepts quickly by simply showing the students a physical version of how we think about the concept without having to use complex mathematics.

Our design of the dimensionality activity illustrates this concept; we use physical blocks to demonstrate *data dimensionality* and *shape*, allowing us to concretely visualize data as a tensor for input into a ML algorithm. As human beings, we typically conceptualize dimensions physically, at least for the first three dimensions. It's easy for us to imagine a dot, a line, a square, and a cube to represent zero through three dimensions. After that, any further dimensions can be imagined as copies of the cube arranged in rows. As ML professionals, when identifying an issue that involves dimensionality, we typically utilize the same type of visualization when approaching a problem. To enable students to similarly approach problems, we once again move away from computers and code to leverage physical materials. In this activity, we give groups of students a

pile of unit-building blocks like the ones used to teach base 10 in grade school. We use the blocks themselves to describe dimensions zero through three, then point out how each classroom table has a large 3-dimensional block; the combination of tables that you can choose from functions as another dimension. Learners then transition into a practice-focused portion of the activity by working in small groups to find a specific item from the blocks on all the tables, which requires understanding how to index blocks as they would with data in code. By directly recreating the structure of thinking we use as more experienced ML practitioners when thinking about data, students were able to leverage their existing knowledge about the physical world and build a useful mental model for how to index and work with arbitrary dimensional data. Here, we relate finding an object, something every human is familiar with, to indexing and data dimensionality, a computer concept integral to understanding Machine Learning – without using a computer in the process. Thus, we use both computer-independent design and recreating our understanding of concepts as student-teachers in the design of this activity.

Enjoyment of learning improves engagement and retention [22], and thus our in-class activities are designed to bring more fun into the classroom while supporting collaborative learning. As an example, a Telephone game activity demonstrates *RNN information loss* without requiring participants to have any knowledge of the tensor storage capacity and information theory that RNN memory loss arises from, while also being fun. Each student is given a singular sentence to remember then asked to stand in a line around the classroom. Similar to the grade school Telephone game, their job is to tell the next person in line their own sentence, as well as everything that was told to them from the previous person. Thus, the 20th student in line will have to remember 20 phrases, which is close to impossible given their length and the time limit. After propagating through the student 'telephone' line, most of the earlier sentences were forgotten or misconstrued, clearly demonstrating how an RNN forgets information. By mimicking the popular game of Telephone, students are engaged in a game that is both illustrative of the issue, enjoyable to interact with, and a break from the typical classroom monotony. Learners are thereby encouraged to participate in class and engage in discussion amongst themselves, thereby facilitating accessible ML education and experiencing benefits to both engagement and retention.

Methods

The primary goal of *Introduction to Machine Learning* is to support the development of industry-needed skills in Machine Learning in an accessible, 10-week university course without prerequisites through an intuition-first, math-second approach. Success would mean the achievement of both theoretical and practical Learning Outcomes (Table 1), evidenced by the learner's ability to create a Convolutional Neural Network or Recurrent Neural Network in PyTorch on an independently-chosen problem during the final project. We measured the success of the course through several key methods. First, a questionnaire at the start of the course prompted students to identify their own exposure to the needed skills and ML tools including Python and foundational math, setting a baseline to evaluate how their skills improved. At the end of the quarter, an anonymized quarterly exit survey prompted students to:

- evaluate the efficacy of the different course features and the classroom environment
- surface confidence in a set of both technical and professional skills
- reflect on the class climate and their sense of belonging

- frame their current thinking in considering an engineering major

This survey is developed in partnership with our university's internal course and program assessment and accreditation body and is administered by them for continuous programmatic improvement. Finally, we inductively analyzed qualitative responses for common themes depicted in the end of the quarter survey and the Student Evaluation of Teaching surveys delivered at the end of the quarter. Thus, all survey data analyzed regarding students' perceptions of this course has already been deidentified and aggregated for anonymity before we gain access to it.

We consider the final project submissions as a holistic indicator of the effectiveness of class instruction. The extent of learner knowledge by the end of the quarter is apparent through the complexity of the final project artifacts, developed and documented in their engineering notebooks. Not only do learners submit their CNN or RNN model on data they find themselves, but they also communicate their understanding and process via a short presentation on their methods during their final exam period, along with a technical writeup that includes data analysis visuals, model performance metrics, and reflection. Additionally, project milestones are assessed every week of the final project leading up to the final presentation with specific key deliverables, giving us insights into individual student's progress throughout their development of the project.

Results

The overwhelming success of the students on their final projects is an effective demonstration of their achieved skills and thus the efficacy of the class instruction. Our class attracts a diversity of early undergraduate students, the largest percentage of which are computer science majors (or proposed computer science majors), as expected. However, computer science students represent only 38% of our enrolled students; in fact, 18% represent majors outside of our School of Engineering. Furthermore, learners in *Intro to ML* are more gender-balanced than the student population in our school of engineering: 39% of enrolled students are female-identifying while the average in our engineering school is 20%. Despite most of these students starting with very little programming experience and typically no ML experience, as evidenced by their responses to the survey prompts visualized in Table 2, every student-team has been able to successfully design, implement, and test their own customized architecture for a convolutional or recurrent neural network on a dataset of their choosing. The project required students to make predictions about their model's outcome, plan out how to build it with block diagrams, perform data analyses, process their data, and finally build and run their model on remote servers using industry tools and services such as Kubernetes or Runpod. Thus, students' effective completion of the final project was a clear demonstration of the following skills which coincide with those recognized as needed by industry professionals [2, 3]:

- Predicting effects of changes on ML structure (Building/Developing ML models)
- Planning an ML project: outlining, research, block diagrams (Developing ML models)
- Processing and analyzing data visually and numerically (Data Analysis)
- Using remote server computation (ML Ops)

The ability of each student enrolled in the class to effectively predict, create, analyze, and deploy either a CNN or RNN ML model and speak competently about their design work demonstrates the development of a holistic understanding of the ML pipeline; this is true across different majors and demographics.

Our anonymized post-class survey, alongside the pre-class baseline-check, found promising results indicating student achievement of course learning outcomes. The data shows gains in student confidence for ML and engineering-related skills, both professional and technical, on questions that asked students to rank their abilities on a Five-Point Likert scale for various skills taught in the class before the course began, and after the course was completed, detailed in Table 2. The staggering jump in self-identified skill proficiencies from completing the course is unexpected for an introductory class (when compared to other classes at our university) and a strong indication that the class's teaching techniques are an effective way of teaching Machine Learning-related skills. Even for secondary outcomes such as "Programming Skills," a large jump can be seen, a significant result for a class with no programming background requirement. These outcomes offer indirect evidence that our class is an accessible way to raise confidence in ML concepts.

Table 2: A comparison of students' confidence in representative course skills taught by Introduction to Machine Learning. These data convey students' self-reported baseline knowledge at the beginning of the course and their perspectives of their skill levels at the end of the course. Note the post-class survey did ask students to reflect back on their skill level at the beginning of the course retroactively.

Number of Students who responded to a survey question with "Very good"/"Excellent" or "strongly agree"/"definitely" which correspond to 4 and 5 on a 5-point Likert scale		
Survey Question	**Before**	**After**
Analyzing data to draw conclusions and determine next steps	14%	73%
Programming skills	33%	68%
Have some kind of ML experience	4%	Not asked
Proficient in Python	19%	Not asked
Complete the Engineering Design Cycle	18%	68%
Collect, process, and/or manage data	0%	59%
Use Statistical Analyses on Data	14%	64%

Questions about class climate in the post-class survey point towards the students enjoying the class and experiencing a supportive and collaborative learning environment. This conclusion comes from descriptive statistics of Likert items targeting sense of belonging in the classroom. Overall, 66% of students responded Yes or Definitely Yes (5 or 6 on 6-point Likert questions) to these series of questions. Table 3 illustrates the responses given by the students for a sampling of these questions, which again show results not typically seen in early engineering classrooms.

Table 3: Selected examples of statements from a series of questions on class climate alongside the percentage of student-respondents who ranked their agreement as a 5 or 6 on a 6-point Likert scale in a post-class survey. These are a sampling of questions asked in this area; the average response to the full series of questions is 66% of students responding 5 or 6 on a 6-point Likert items.

Number of Students who responded to a survey question with "agree"/"yes" or "strongly agree"/"definitely" which correspond to 5 and 6 on a 6-point Likert scale	
Survey Question	**% of Student-Respondents**
In general, I feel welcome in the classroom.	95%
In general, I feel supported during class time.	82%
I can be myself during class time.	77%
I would feel comfortable asking the student-instructor for help (via email or in person) if I did not understand the course material.	59%
A student can ask a question or give a wrong answer without feeling judged by other students in this class.	73%

Furthermore, students from the first cohort (Winter 2025) continue to pursue various ML paths after taking this course. Two students from the class joined two different ML labs, one to work on LLMs and the other to develop ML medical imaging techniques. Another student joined the school's AI club to lead workshops on creating Neural Networks. Yet another became the president of our University's Association for Computing Machinery club. A PhD student-researcher from one of these ML labs commented about one of the undergrads who had taken our course joining their lab *"the student was exceptionally skilled in ML techniques"* and was *"surprised by their knowledge, abilities, and experience in remote server model training."* Finally, two other students specifically credit their work in the class as helping them gain professional internships.

Next Steps

The course is set up to allow students two options as to what model they wanted to make for their final project: Recurrent Neural Network (RNN) or Convolutional Neural Network (CNN) depending on the problem they chose to solve and the data they chose to analyze. We cover both in detail during class, both in terms of theory and code. However, we quickly discovered an issue: that the majority of sequential datasets are text data. We know from experience that text data are incredibly difficult to work with, as it requires a significant amount of preprocessing in the form of tokenization, and takes a lot more time and computation to generate a good model. Because of this, few teams select this option for the final project, likely due to its more complicated nature - especially when it comes to coding. Therefore, we are considering to focus the class on CNNs and slightly narrow the breadth during our next iteration of the course.

Another area for improvement is around how we present and grade engineering notebooks. Documentation turned out to be a major point of tension for the learners as properly utilizing the notebooks is not trivial to understand and can feel tedious if the benefits are not understood. As notebooks are submitted for a grade, many students lose sight of the benefits of documentation, missing the critical thinking, predicting, planning, and reflection that proper documentation encourages; activities that not only helps students better learn the material, but are themselves pivotal skills to learn before they enter the workforce. We are currently collecting data on a new activity that hopefully addresses these difficulties experientially by actively demonstrating the benefits of good documentation in a memorable way.

More pressingly than any specific issues, our approach as a whole still has limitations that should be addressed going forward. For one, many of the skills taught involve coding and working with digital data, which makes it difficult to teach some of the more practical skills in a computer-independent classroom activity. One option which we've already planned for our next iteration of the class, is to teach the students how to convert block diagrams to code, allowing them to build their ML models physically with blocks represented by something tangible, such as pieces of paper, and then only involving the computer to convert the blocks to code. This approach could help learners focus on the logic of the model by emphasizing planning. While the need for a computer isn't completely removed, block diagrams are possibly the closest you can get to coding ML models in a computer-independent way. We've designed a lesson to teach LSTMs as the next step up from RNNs using cut-out pieces of paper and will test its effectiveness in the upcoming iteration of the course. Using an LSTM to show how to code arbitrary architectures, however, is simply one of many coding skills that we would like to move away from computer-based activities. Many other ideas such as Python classes, training loop code, PyTorch documentation, and base Neural Network architecture code would benefit from being reworked away from an activity that requires a computer.

Conclusion

The current state of Machine Learning education inhibits students from learning what they need to succeed in the job market and real life. While market analyses are showing an underrepresentation of much needed ML skills, university curricula continue to gatekeep Machine Learning courses behind prerequisites and math understanding that stop students from accessing them until their final years of college - and by then many initially enthusiastic students may have moved to other, better supported, fields. Additionally, students from other disciplinary backgrounds are completely excluded from learning about an ever-growing aspect of modern life. Many online resources try and fail to pick up this slack due to their passive and unstructured nature, leaving students to seek other extracurricular experiences with limited resources.

Intro to Machine Learning addresses these issues with an activity-focused, high-structure class framework that guides students through intermediary activities, skill-building assignments and projects supported by incremental milestones that keep learners on track. Experiential learning and high-structure allow us to quickly teach skills and topics through an intuition-first approach, developing a foundation of holistic understanding that may then be built upon. In this way, learners are able to interact with ideas in a low-risk environment, and with incremental deadlines,

learners new to the field have ample practice and preparation to complete a complex final project with learning outcomes previously unheard of for an introductory, 10-week university course.

Our experiential activity design follows several key design patterns; the three seemingly most impactful ones are: computer-independent participation, developing activities around our own mental models, and creating enjoyment for the students. Removing the computer from activities allows us to better engage students and utilize their existing knowledge of the real world. Designing activities around our mental models means we create useful paradigms for students to think about complex topics, ones that we ourselves as students in ML professions use. Finally, enjoyable participation in the classroom helps create a collaborative environment that supports learners in interacting with complex topics without fear of failure. Overall, intuitive activity design brings far greater accessibility to the course topics, opening the door for many students who otherwise would not be able to learn these skills. The effectiveness of our approach is seen through the collaborative final project completed during the final weeks of the course. Students plan and design a problem, then in successive milestones process data, perform data analysis, plan model architecture, train their model, and reflect on their decision making. All student teams have been successful in creating a CNN or RNN model for a problem of their choosing, displaying mastery over the learning outcomes in the class as well as their use of soft skills such as collaboration and documentation. An anonymous end of course survey depicts these successes: student-responses show a great increase in skills and understanding of the material, as well as their enjoyment and comfortability in the classroom thanks to the low-risk, collaborative learning environment.

References

[1] D. Bergmann, "What is machine learning?" IBM. [Online]. Available: https://www.ibm.com/think/topics/machine-learning, accessed: Feb. 22, 2026.

[2] M. Dangelo, "Needed AI skills facing unknown regulations and advancements," Reuters. [Online]. Available: https://www.thomsonreuters.com/en-us/posts/technology/needed-ai-skills/, Dec. 2023, accessed: Jan. 21, 2026.

[3] F. Jaumotte, J. Kim, D. Koll, E. Li, L. Li, G. Melina, A. Song, and M. M. Tavares, "Bridging skill gaps for the future: New jobs creation in the AI age," *IMF Staff Discussion Notes*, vol. 2026, no. 001, Jan. 2026.

[4] Glassdoor, "Machine learning engineer salaries," [Online]. Available: https://www.glassdoor.com/Salaries/machine-learning-engineer-salary-SRCH_KO0,25.htm, accessed: Jan. 21, 2026.

[5] Glassdoor, "Frontend developer salaries," [Online]. Available: https://www.glassdoor.com/Salaries/frontend-developer-salary-SRCH_KO0,19.htm, accessed: Jan. 21, 2026.

[6] Glassdoor, "Backend developer salaries," [Online]. Available: https://www.glassdoor.com/Salaries/backend-developer-salary-SRCH_KO0,17.htm, accessed: Jan. 21, 2026.

[7] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, no. 1, pp. 147–169, 1985.

[8] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[9] K. I. Albanese, S. Barbe, S. Tagami *et al.*, "Computational protein design," *Nature Reviews Methods Primers*, vol. 5, p. 13, 2025.

[10] E. Brooks, K. Glemaud, and R. Morse, "Best artificial intelligence programs," U.S. News & World Report. [Online]. Available: https://www.usnews.com/best-graduate-schools/top-science-schools/artificial-intelligence-rankings, accessed: Jan. 21, 2026.

[11] K. Jordan, "Initial trends in enrolment and completion of massive open online courses," *The International Review of Research in Open and Distributed Learning*, vol. 15, no. 1, Jan. 2014. [Online]. Available: https://www.irrodl.org/index.php/irrodl/article/view/1651

[12] J. Reich and J. A. Ruipérez-Valiente, "The mooc pivot," *Science*, vol. 363, no. 6423, pp. 130–131, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aav7958

[13] T. Favaloro, "Building the engineering identity of the lower-division engineer: A formal model for informal peer-to-peer mentorship and student leadership through undergraduate student-led experiential learning," in *2024 ASEE Annual Conference & Exposition Proceedings*. Portland, Oregon: ASEE Conferences, Jun. 2024, p. 48430.

[14] T. Favaloro, "Students as the teachers: Positioning undergraduates as experts, role-models, and guides to create diverse learning communities," in *IEEE Frontiers in Education Conference (FIE)*, Washington, DC, USA, 2024, pp. 1–9.

[15] E. Wachtel, Q. Cao, M. Kaltman, K. Tran, M. R. Hernandez, and T. Favaloro, "Board 174: Fostering inclusivity and engagement while learning by doing: A new paradigm in engineering education based on student-designed, student-taught courses," in *2024 ASEE Annual Conference & Exposition Proceedings*. Portland, Oregon: ASEE Conferences, Jun. 2024, p. 46737.

[16] I. H. Phan, I. Taranenko, and T. Favaloro, "Hacking the system: A peer-led cybersecurity course for early-career university students," in *2025 ASEE Annual Conference & Exposition Proceedings*, 2025.

[17] D. Yaralioglu, Y. Wang, K. Lin, and T. Favaloro, "Teaching creative design in virtual reality: A course designed and taught by students, for students," in *2025 ASEE Annual Conference & Exposition Proceedings*, 2025.

[18] S. Freeman *et al.*, "Active learning increases student performance in science, engineering, and mathematics," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8410–8415, May 2014.

[19] E. J. Theobald *et al.*, "Active learning narrows achievement gaps for underrepresented students in undergraduate science, technology, engineering, and math," *Proceedings of the National Academy of Sciences*, vol. 117, no. 12, pp. 6476–6483, Mar. 2020.

[20] D. C. Haak, J. HilleRisLambers, E. Pitre, and S. Freeman, "Increased structure and active learning reduce the achievement gap in introductory biology," *Science*, vol. 332, no. 6034, pp. 1213–1216, Jun. 2011.

[21] J. M. Lang, *Small Teaching: Everyday Lessons from the Science of Learning*. Hoboken, NJ: John Wiley & Sons, 2016.

[22] D. A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development*, 2nd ed. Pearson FT Press, 2015.