Teaching Statement: Machine Learning Teaching Needs a Full-Stack Approach

Huan Zhang

Teaching Philosophy My goal in teaching machine learning is to ensure that students have the curiosity to explore every part of machine learning and understand every aspect of how their models run — no black boxes. To achieve this goal, I advocate that **machine learning teaching needs to bring in more "full-stack" elements**, connecting materials from different courses ranging from software to hardware to give students a holistic understanding of machine learning.

My teaching and learning experiences show that machine learning courses are often developed separately from other computer science courses, unlike the traditional curriculum which is often well connected from both software and hardware sides. Consequentially, students often **fail to make connections between machine learning and other courses and cannot apply what they learned**. For example, in machine learning classes, a student may be taught about convolutional neural networks (CNNs). This topic has a few key elements: (1) Why is convolution effective in computer vision? (2) How to use a deep learning framework to implement it? (3) What are the underlying algorithms used to compute the convolution? (4) How do the algorithms efficiently run on accelerators such as GPUs?

Unfortunately, many machine learning courses answer only the first two questions. The latter two questions require knowledge of algorithms, compilers, operating systems, and computer architecture. Although students may have taken those courses, they often do not realize their connections to machine learning. As a result, students have the impression that they only need to code their model superficially, and everything else happens magically in a black box. During a one-on-one coding session with a student I mentored to resolve the problem of slow training code (she already has some experience in deep learning), she once asked me a question "what does the reported GPU utilization mean when I train my models"? If she understands better how her code runs, it will be much easier to find the bottlenecks in her program.

When I was a teaching assistant for a Big Data course with many programming assignments of machine learning algorithms, students greatly appreciated my efforts in explaining how Python interpreter works and why their for-loops are slow in the discussion sessions. If I teach a deep learning class, to fully address the four questions mentioned above, I will include discussions on how convolutions are effectively performed using implicit matrix multiplications, how parallel matrix multiplication algorithms work, and how GPUs are designed the efficiently accelerate this type of workload. I will spend at least one lab session asking students to trace deeply into a simple deep-learning software framework, open the black box, and see how each part is connected to what they have learned in this and other courses.

There are multiple benefits when taking this "full-stack" approach, which also align with **core parts of my teaching philosophy**. First, it **stimulates the curiosity** of students to open all the black boxes in the topic they learn. Second, it encourages **learning by doing** and gives students **hands-on experiences** in different disciplines of computer science, allowing them to connect better what they know to practical problems in machine learning. Third, it encourages **critical thinking** because students learn to understand the course material holistically and can take the same approach when encountering new problems. Last but not least, it can **encourage independent research** because students often realize parts of the system can be improved or further studied during this unboxing process.

Finally, I also believe that it is essential to **teach ethics** in machine learning and artificial intelligence (AI) courses, because AI can be misused due to its power. My research overlaps with AI trustworthiness, and I am aware of many societal and ethical issues in AI, such as fairness, privacy, transparency, and diversity, which can support my teaching.

Teaching experience I gained teaching experience via teaching assistantships, guest lectures, and tutorials.

- Teaching assistantships: I served as a teaching assistant for **three undergraduate courses** in quite different perspectives of computer science: "Parallel Computer Architecture" in Spring 2013 (about 50 students), "Probability and Statistical Modeling for Computer Science" in Fall 2015 (about 200 students), and "Big Data & High-Performance Statistical Computing" in Spring 2017 (about 50 students).
- Guest Lectures: I gave **three guest lectures** at different institutions, including UIUC (CS 562: *Advanced Topics in Security, Privacy and Machine Learning*), Stony Brook University (CSE 510: *Hybrid Systems*), and University of Nebraska

Lincoln (CSCE 990: *Deep Learning and Assured Autonomy Analysis*). These lectures were about my research on neural network verification and were prepared for students with basic machine learning backgrounds.

- Tutorials: I lead the development of a two-hour tutorial on state-of-the-art research of neural network verification (available at `neural-network-verification.com`) with my collaborators. The tutorial aims to introduce the background of neural network verification and a few representative algorithms, as well as give hands-on programming demonstrations to allow practitioners to apply verification to their applications. The tutorial contains videos and interactive Colab coding examples. I presented the tutorial with my collaborators at AAAI 2022, and part of the tutorial at Lorentz Center Workshop on Robust Artificial Intelligence in 2021.

Teaching Interests and Example Courses My interdisciplinary research has prepared me to teach a variety of courses on different topics, not limited to machine learning and artificial intelligence. Particularly, my work aims to address the challenges of AI safety in computer security, and my verification algorithms for AI are inspired by techniques in optimization, programming languages, and computer-aided verification; in addition, my expertise in parallel computing and computer architecture was crucial to scale up my algorithms. My teaching plan specifically includes:

- Undergraduate-level courses: Common courses include introductory courses for *Machine Learning*, *Artificial Intelligence*, *Computer Security*, *Programming Languages*, *Data Science*, *Optimization*, *Computer Organization* and *Computer Architecture*. For machine learning courses, I plan to revise their curriculum to include “full-stack” case studies intersecting with different disciplines in computer science, as I advocated above in Teaching Philosophy.
- Graduate-level courses: Common courses include *Optimization*, *Computer Vision*, and graduate-level machine learning courses. I also plan to develop a few graduate-level classes: *Formal Verification of Artificial Intelligence* covers my field of study, with an introduction to formulation of the formal verification problem in AI and a study of verification algorithms for different types of AIs; *Trustworthy Machine Learning* is a seminar class covering a broader range of topics not limited to AI security and verification, but also fairness, privacy, transparency, ethics and other trustworthiness aspects of AI. These courses are also research-oriented and I encourage students to engage in relevant research projects.

Mentoring Experience I directly mentored ten undergraduate students and five junior graduate students during my graduate studies and postdoc. My interactions with them have been productive: I co-authored nine papers with undergraduate, master or junior PhD students. In the α, β-CROWN neural network verifier project which I am leading, four junior students have contributed significantly to the development this open-source software.

My goal in mentoring students is to **stimulate their interest in conducting research** and support them to **grow up as mature and independent researchers**. Since students may have different levels of background, I design projects with appropriate difficulty levels and ensure that students do not get stuck initially and can also always learn and grow. For students with less experience, I typically assign them very specific projects with well-defined goals, such as implementing a function in the α, β-CROWN verifier. When they gain more experience, I assign them one specific research idea that they need to explore independently and publish. Finally, when students become more senior, I encourage them to choose research problems independently and help them sharpen their own research ideas. During the process, I also aim to build their **communication skills**, and **resilience** so that they can remain calm even if some research ideas fail.

An important resource for my mentorship is the α,β-CROWN neural network verifier, which is under active development. It is a good platform for students to learn the backgrounds of neural network verification, and they also get a chance to get involved in the development of state-of-the-art algorithms and tools. Students often get excited when seeing their contributions publicly acknowledged in a state-of-the-art toolbox used by many researchers.

As an example, Zhouxing Shi, an undergraduate student I mentored three years ago, was involved in the development of my neural network verifier in the summer of 2019. Zhouxing got interested in the research of neural network verification and eventually became a PhD student with my advisor at UCLA. Zhouxing’s research is quite successful as a junior PhD student – he published **four first-author papers in top-tier conferences** under my mentorship. Other students with similar experiences include Yihan Wang (female) and Li-Cheng Lan at UCLA.

Conclusion To conclude, I am dedicated to improving machine learning teaching with new insights during my career, including the full-stack approach discussed above. I am also committed to fostering talented students and turning them into independent researchers, and I look forward to seeing their success.