TEACHING STATEMENT

Teaching Students to Reason, Build, and Evaluate Trustworthy AI

Enyi (Olivia) Jiang

enyij@stanford.edu · enyij2@illinois.edu · enyijiang.github.io

My goal as an educator is to help students become independent technical thinkers: able to explain why a method works, implement it carefully, and recognize when the evidence does not support a claim. In computer science and engineering, these abilities connect mathematical foundations to the responsible design of real systems. I want students to leave my courses with the confidence to build and the judgment to question what they have built.

My preparation spans six courses at the University of Illinois Urbana-Champaign: Discrete Structures, Data Structures, Probability and Statistics for Computer Science, Natural Language Processing, Trustworthy AI Systems, and Applied Machine Learning, for which my teaching assistant appointments include Spring and Fall 2026. I have also mentored undergraduate and master's students at UIUC and Stanford and serve as a mentor for the Red-teaming/Adversarial and Interpretability streams of AI Alignment @ UIUC. These experiences provide the foundation for the teaching and mentoring approach I will bring to a CS/ECE department.

1. Connect mathematical ideas to implementation and evidence

My teaching will connect three forms of understanding: a precise statement of a concept, a working implementation, and an explanation of what the implementation demonstrates. My TA experience across discrete mathematics, data structures, probability, and machine learning gives me a foundation for helping students make these connections across the curriculum.

For example, in an applied machine learning course, I would introduce generalization through a small prediction task. Students would first specify a train–test split and explain what it is intended to measure, then implement a baseline, and finally change the test distribution. A short written analysis would ask why a strong initial score may not predict performance in the new setting. The objective is to connect probability, experimental design, and programming through a question students can investigate directly.

In foundational courses, I would use the same progression at a different level. A data structures exercise could ask students to predict an algorithm's behavior, trace a small example, and explain how an invariant supports correctness. Students would then test an edge case that challenges their initial intuition. This gives both a concrete entry point and a reason to engage with formal arguments.

2. Make reasoning visible and feedback actionable

A correct final answer gives an incomplete picture of student understanding. I will use short explanations, code walkthroughs, and error analysis to make students' reasoning visible. Before introducing a solution, I would ask students to make an individual prediction, discuss it with a peer, and identify which assumption explains any disagreement. Their responses would help me decide whether to revisit the concept or proceed to a more demanding application.

Assignments will separate mathematical reasoning, implementation, and interpretation in their rubrics. In a machine learning project, students should receive credit for a well-designed comparison and a justified conclusion

even when an ambitious method does not outperform the baseline. Staged submissions—a question and baseline, an experimental plan, and a final analysis—will make feedback useful before the project is complete. I will use brief concept checks and anonymous midsemester feedback to identify recurring difficulties and adjust instruction.

3. Support students with different preparation and goals

Students enter CS/ECE courses with different levels of mathematical confidence and programming experience. I will make prerequisites and expectations explicit, provide concise review materials, and offer low-stakes exercises that help students identify gaps early. Core assignments will target shared learning goals, with optional extensions for students ready to go further. Written questions, small-group discussion, and individual explanations will provide multiple ways to participate.

For team projects, I will ask students to document contributions and rotate technical responsibilities so that every member practices implementation and interpretation. Clear milestones and predictable feedback will help students plan their work and seek support early.

For AI-assisted work, I will state permitted uses, require disclosure where assistance is allowed, and ask students to explain and test their submissions. Some exercises will emphasize unaided reasoning; others will examine how to validate an AI-generated solution. Both should develop intellectual ownership of the result.

4. Mentor students toward ownership of research questions

I have mentored six undergraduate and master's students across UIUC and Stanford and mentor in AI Alignment @ UIUC. I want to help students move from reading a paper to formulating a question they can investigate independently.

In my laboratory, I will structure early projects around a manageable sequence: reproduce a result, identify an assumption, and design a test that could change the conclusion. For an interpretability project, for example, a student might first reproduce a probe, then ask whether its predictions persist under a controlled intervention. This creates a concrete contribution without requiring a new student to solve an entire research problem at once.

I will establish expectations about scope, meetings, documentation, and authorship early. Regular discussions will distinguish implementation obstacles from conceptual uncertainty and progressively transfer decisions to the student. My aim is for mentees to learn how to choose the next experiment, explain a negative result, and articulate their contribution, while supporting their individual career goals.

5. Contribute across the CS/ECE curriculum

I am prepared to contribute to undergraduate teaching in data structures, discrete mathematics, probability and statistics, and machine learning, as well as advanced courses in natural language processing and trustworthy AI. My computer engineering background also helps connect learning algorithms with uncertainty, reliability, and computational cost.

I would develop a graduate course, *Mechanistic AI Safety: Evaluation, Alignment, and Intervention*, organized around evaluating claims rather than only surveying papers. Students would reproduce a safety or interpretability result, design a controlled stress test, and evaluate an intervention under a modest compute budget. A final report would distinguish observations, causal evidence, and remaining uncertainty. Across foundational courses and research mentoring, I want students to develop the same lasting habit: make a precise claim, seek the evidence needed to test it, and revise their understanding when that evidence changes.