Eve Fleisig
Teaching Statement

I love teaching and mentorship with a passion. I've mentored ten highly successful students during my PhD and was a primary instructor for Berkeley's 200-person Introduction to AI class (CS 188), as well as TA'ing for more advanced courses on machine learning and NLP. During my PhD, I earned Berkeley's Outstanding Graduate Student Instructor Award and the Evergreen Award for undergraduate mentoring. I'm enthusiastic about continuing to mentor students, refresh curricula, and teach classes across a range of areas, class sizes, and difficulty levels.

Teaching philosophy. To me, teaching computer science combines two of the most fun things to teach: engaging mathematical and logical problems (How do I search this graph quickly? How do I write these loops efficiently?) and fundamental questions about technology that affect society at large (How do we build a machine that understands language? What happens if we embed that system in an existing societal process, like automating clinical notes?). Teaching computer science presents the challenge of instilling the thrill of solving neat logical problems, and expanding that consciousness to tackling multifaceted, difficult research questions with many potential answers.

In practice, this is no easy task. We often teach very large classes with students at very different levels of previous experience, many already stressed or overworked. In my teaching so far, I have aimed to create a haven where students at different degrees of experience and engagement discover a new understanding and renewed enthusiasm for the material. I want a student in an introductory class who's struggling with computer science to feel supported and ready to take on the challenge. I want a student who's good at coding but hasn't found a passion for it yet to go home excited about what computer science can do. And I want someone already engaged with machine learning to come home with new questions and ideas to shape the future of the field.

Teaching computer science with clarity and rigor is vital; but so, too, is engaging with students and inspiring them to pursue it after they close their exam books. When teaching, I aim to take that extra step, which fosters greater understanding of the material and plants the seeds for future researchers, founders, and engineers.

Fostering engagement in large and small classes. In undergrad, I was the sole TA for Princeton's NLP research seminar. Students were expected to produce projects that required NLP skills, but most had little experience with the field; I ended up essentially tutoring them in office hours on how to train and evaluate language models. I found it was also a valuable opportunity to connect them with research relevant to what they were interested in and explain concepts they hadn't yet studied, from nuances of model training to the internals of transformers.

At Berkeley, I've relished the challenge of teaching much larger undergraduate classes in machine learning. I was a teaching assistant for CS 189, a math-heavy machine learning class, and co-taught CS 188, a large introductory machine learning class. When lecturing to 200 students, keeping all of

them engaged and making sure they understand the material is a challenge. One strategy I used was to do two major passes right before each lecture: one to identify parts of the material that were likely to confuse students, and go over the problems to ensure they're crystal-clear, and one to make sure there are interludes drawing connections to modern applications and interesting challenges between those sections to keep students engaged. I think keeping students engaged is a multifaceted process: keeping lectures easy to follow; ensuring constant questions and feedback from students in lecture; being accessible in office hours and ready to show students how their questions link to broader research in the field; monitoring assignment difficulty so students are challenged but not overwhelmed; and communicating frequently with TAs to make sure logistics go smoothly.

I've also guest lectured for several classes at a wide range of levels—from introductory CS classes to Berkeley's graduate NLP course—on machine learning, AI, and/or its ethical issues. I introduced new lectures on these topics to CS 188, Berkeley's introductory machine learning class. As a result, I'm experienced with tuning this content to suit a range of audiences. I'm also interested in further developing curricula in AI ethics that can either stand alone or be integrated into existing classes on machine learning.

Strengthening assignment design. Engaging lectures are important, but a student's main stressors are homework and exams, which are key to cementing their knowledge, form much of their impression of the class, and influence whether they stick with computer science. I supervised exam design for CS 188, and designed multiple assignments, exam questions, and discussion problems for CS 188 and 189. For fun, I'm also a writer and head editor of academic trivia questions for national competitions (including heading an acclaimed question set that won Tournament of the Year), so I've thought long and hard about what it means to write questions that deeply test knowledge of the material, and separate students with different levels of understanding, without being excessively punitive, confusing, or simplistic. Beyond the content itself, creating questions that are clear, challenging, and reward deep knowledge of the material relies on having a reliable procedure for writing a strong exam or problem set: coordinating desiderata with question writers, testing thoroughly for difficulty and clarity, and reviewing for issues. I quite enjoy running this kind of process: both writing questions myself, and seeing a polished, fair, and engaging product come out of writing, discussion, and feedback with TAs.

That communication also helps when unexpected challenges occur: in CS 188, our final exam had to pivot online with only a few hours' notice due to a campus gun threat. We quickly revamped our procedures, coordinated announcements to students, and set up online monitoring at short notice. Because we had prepped well in advance, we were able to ensure everything ran smoothly.

Mentoring. Mentoring students is extremely important to me. I mentored ten undergraduates over the course of my PhD, and supervised two of them in subsequent Master's theses, establishing a mini-research lab. Many of these collaborations resulted in research papers, typically first-author papers for the students, including one that earned an EMNLP paper award. I

also frequently meet with undergrads interested in research even if I don't have bandwidth to work with them, and have been delighted to see several of them go on to graduate school in computer science.

I particularly enjoy helping students grow from a cautious interest in an area to full-fledged enthusiasm. For students I've worked with over longer periods of time, I especially like helping to transition them from well-defined, curated projects I've set up, gradually increasing flexibility, to projects they can design and lead themselves. For example, my student Kayla Lee came in to help with a project on RLHF; developed an interest in task-oriented dialogue that I encouraged through research meetings, guidance on open research questions, and code review; and is now a YC-funded founder for a task-oriented dialogue agent startup. My student Vyoma Raman came in interested in difficult questions about how to measure complicated qualitative bias issues; we worked together to refine her ideas on fairness evaluation, and she's now pursuing a PhD in that direction. Another of my students, Samuel Ghezae, began research with much enthusiasm but without NLP experience and with a short summer timeline for research; I put together an NLP crash course for him and he subsequently completed a project on low-resource language modeling. He's now applying to PhD programs in machine learning.

When mentoring students, I try to communicate frequently in order to understand what they need: high-level mentorship, discussion of research ideas, and fostering a love of research; balanced with assistance on individual problems, making sure they don't get stuck in the weeds, and strengthening their technical abilities. By gradually scaling up project difficulty and independence, I've been proud to see these students grow as people and as researchers. They have gone on to attend top graduate programs such as Stanford and CMU, become startup founders, and join top industry labs such as OpenAI and Anthropic.

Future plans. I envision bringing on students with a relatively wide range of interests related to natural language processing and/or fair and responsible AI. Having worked with graduate students and undergrads in a range of labs, including in theory and systems, and collaborated across departments, I'm interested in co-advising students between different areas of computer science or with related areas such as linguistics. One thing I aim to establish is a supportive group culture that makes students feel prepared to do their best work: via group meetings, explicit mentorship between senior and junior PhD students, and ensuring that students meet with me regularly both for project progress and their broader PhD goals.

I look forward to teaching a range of core classes on machine learning, artificial intelligence, and natural language processing, as well as running smaller seminars or co-designing curricula related to ethical AI and responsible NLP. I am enthusiastic about teaching both large and small classes, advanced and introductory. Each of these brings its own challenges that I enjoy taking on.