Reflections on Teaching

Nirav Atre, *Carnegie Mellon University*

Teaching (both inside and outside the classroom) has been one of the defining highlights of my graduate school experience. To me, the prospect of educating, mentoring, and advising the next generation of scientists is a *deeply fulfilling* one, and a major reason behind my decision to pursue an academic career. In what follows, I will outline my teaching philosophy (i.e., the key pedagogical lessons I have learned over the years), and how I have applied it both in the context of classroom teaching and as a research mentor.

Classroom Teaching

As a Ph.D. student at CMU, I served as a **teaching assistant (TA) for 3 undergraduate and master's level courses**. One was a distributed systems course taken by a large fraction of CS undergraduates (200+ enrolled students), and the other two were successive iterations of a networking course (60-90 students). For each of these courses, I helped develop and release a major course project, delivered lectures, prepared recitations, held weekly office hours, and graded assignments. I also served as the Head TA for one iteration of the networking course, which expanded my responsibility to course logistics (*e.g.*, TA task delegation). Finally, I **co-instructed a doctoral-level networking course** with my advisor, Prof. Justine Sherry, during which I was involved in every aspect of the course, right from designing the course curriculum to evaluating semester-long, open-ended research projects which the students undertook. Coincidentally, my advisor was forced to go on maternity leave mid-way through the semester, giving me the opportunity to "run things" by myself. This was a transformative teaching stint for me, because it gave me first-hand experience dealing with issues (*e.g.*, discretionary changes to deadlines, disability accommodations, identifying students who were struggling) which I would otherwise be insulated from as a TA.

I am most qualified to teach undergraduate and graduate level networking courses. I can also teach introductory computer science courses (*e.g.*, programming fundamentals, algorithms and data-structures), and systems-adjacent topics (*e.g.*, distributed systems, operating systems). I am particularly interested in developing a graduate-level course related to my dissertation work: identifying, understanding, and mitigating *model incongruities*, which are scenarios where our mental and/or mathematical models of systems diverge from reality. Computer security courses teach students how incongruities manifest as *security vulnerabilities* that an attacker can exploit. However, as I explain in my research statement, incongruities can cause systems to misbehave in ways that affect properties beyond just security (*e.g.*, performance, scalability), and in many different contexts (*e.g.*, cyber-physical systems, machine learning). To that end, this course would teach students (via seminar-style research readings) *where* and *why* incongruities arise in modern systems, followed by an open-ended project exploring model incongruities in a research area of their choosing. There are three objectives I hope to achieve with this course: (a) inculcating good reading habits, (b) improving students' ability to reason about systems and their models thereof, and (c) encouraging collaboration and multi-disciplinary research by bringing together graduate students with diverse backgrounds and interests.

Research Mentorship

Over the course of my Ph.D., I have had the immense pleasure of mentoring several junior Ph.D. students in my research group, as well as two incredibly talented undergraduate research interns. My mentorship approach reflects an advising style that I deeply admired in (and borrowed from) my own advisor: *play to individual students' strengths.*

Benjamin Carleton was a summer REU intern at CMU who was interested in exploring *delayed hits* [4], an unusual phenomenon affecting modern caching systems. I realized early on that Benjamin was a gifted hacker, and that he would benefit most from hands-on systems experience. In the short span of 4 months, he built, tested, and evaluated a simulator to model delayed hits in multi-tier caching systems. He published his research [5] as a poster at ACM SOSP 2021, a top systems conference, and was awarded the first prize in the undergraduate Student Research Competition (SRC) for ACM SIGOPS. He is now a Ph.D. student in Computer Science at Cornell University.

Erica Chiang was an undergraduate student at CMU whom I worked with for over 1.5 years. When we first recruited Erica, it was with the intention of helping me build the SurgeProtector scheduler in software. She was in fact a strong programmer, but what *really* struck me about Erica was the mathematical maturity that reflected in her questions about the scheduling algorithm. We quickly pivoted her project to explore a more theoretical line of reasoning (analyzing the robustness of job size heuristics used in SurgeProtector), at which she absolutely *thrived*. She was a co-author on our paper [3] that appeared at ACM SIGCOMM 2022; she presented a poster on her own research [6] at the same conference, for which she won the second (runner-up) prize in the undergraduate SRC. She was subsequently awarded the NSF Graduate Research Fellowship (GRFP), and she is also currently a Ph.D. student in Computer Science at Cornell University.

Teaching Philosophy

My teaching philosophy is student-centric: as the most important stakeholders of the education system, students deserve to be taught the *transferable skills* they need in a manner that is *grounded in educational research*, efficiently utilizes their time and cognitive capacity, and is *aligned with their strengths* as individuals.

Hand out tools, not answers. My own experience as a student has convinced me that specific concepts are easily learned, and just as easily forgotten. Instead, it's the "conceptual toolkit" we build in our scholastic life that helps us solve not just today's course assignments, but also tackle the novel problems we encounter years later in our professional careers; to that end, I strongly believe that my responsibility as an educator is to *cultivate skills and knowledge to address a broad class of problems beyond the classroom.* In the context of course development, I employ outcome-based learning [1], where content is centered around objective and measurable learning outcomes. As an instructor, my goal is then to ensure that the learning outcomes I define are specific enough to be cultivated in the time-frame of one semester, but general enough to hone transferable skills that can be applied beyond it. I use this as a guiding principle in creating new lecture slides, course assignments, and recitation material. For instance, in developing a new course project for the networking course [2], I meticulously abstracted away various nuances of the C programming language and the Linux network stack to help students focus on the concept at hand: designing a loop-free network routing protocol. In the context of student interactions (*e.g.*, during office hours), I always prioritize teaching students how to apply general tools and techniques instead of answering one-off questions; for instance, I always start debugging sessions with a brief tutorial on *gdb* or *valgrind* instead of jumping into code.

Build on common ground. As a student, when I found myself lost in a lecture, it was rarely because the instructor lacked oratory skills or had unintelligible handwriting; rather, it was because the instructor assumed I knew more than I actually did. My takeaway from such experiences is that, while quality of instruction is certainly important, a bigger impediment to learning is the so-called *expert blind spot*: the instructor, being an expert in their area, fails to relate to someone who isn't. As an instructor, my strategy to counteract this is to *start with something the student already knows, and use this as a conceptual building block to bootstrap the learning process.* In the classroom, this typically corresponds to connecting course concepts to everyday phenomena that students can relate to. For instance, I teach pipeline parallelism (a non-trivial concept for undergraduates who have not taken a computer architecture course) by relating it to two housemates who both want to do laundry on a Sunday night.[1] Similarly, in my interactions with students during recitations and office hours, I always strive to establish a common knowledge baseline by first asking them to describe what they understand about the topic, and framing my answer accordingly. For instance, in a single distributed systems office hour, I've had two very different discussions regarding deadlocks in students' Golang code: one with a student who had a simple typo in their instantiation of a Go channel, and another with a student who had a fundamental misunderstanding about their usage. Finally, since many students solidify their learning by teaching rather than listening, I will ask students to describe, in their own words, what they have learned/understood, thereby also *ending with what they know.*

Play to students' strengths. Students have diverse strengths: some are natural theoreticians, while others are gifted hackers. Nevertheless, I strongly believe that anyone can learn anything; the real question is *how*. In my experience, the most effective way to convey information is by tailoring the learning experience to them (i.e., playing to students' individual strengths). While this is challenging to implement in the classroom setting, my primary goal in smaller groups and one-on-one meetings is to *understand how best to explain a certain concept to the student.* In some ways, this is facilitated by their initial explanation of the concept: if they use math and symbols, so do I; if they draw a graph, then I frame my explanation in the context of their graph. I find this strategy to be particularly helpful with my research mentees: working closely with a student gives me ample opportunity to understand their strengths and preferences, and to adapt my mentorship style to their needs. For instance, I realized that Erica, despite

[1]Instead of waiting for the first housemate, A, to finish their laundry, the second housemate, B, can start her washer cycle as soon as A moves her clothes to the dryer. Students quickly see that, with many housemates, this takes about half as long as naïvely doing laundry one after the other.

having taken mostly programming courses, had a strong analytical intuition; correspondingly, we pivoted to a more theoretical direction for her research, at which she was able to excel.

Conclusion

I am fortunate to have had some truly remarkable instructors, mentors, and research advisors over the years, all of whom have had an indelible impact on me, not only as an engineer and a researcher, but also as a person. The potential and privilege to have this kind of impact on many generations of scholars to come is what drives me to pursue an academic career at a university. In terms of classroom instruction, I believe in using research-backed teaching strategies to hone transferable skills that will continue to benefit students well beyond their scholastic years. In terms of mentorship, I am committed to playing to my students' strengths, giving them all the advice, support, and intellectual freedom they need to thrive.

References

[1] Stephen Adam. An introduction to learning outcomes, 2006.

[2] Nirav Atre. 15-441 Project 1: Mixnet. https://github.com/computer-networks/cmu-mixnet, 2021.

[3] Nirav Atre, Hugo Sadok, Erica Chiang, Weina Wang, and Justine Sherry. SurgeProtector: Mitigating Temporal Algorithmic Complexity Attacks using Adversarial Scheduling. In *Proceedings of the 2022 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, New York, NY, USA, 2022. Association for Computing Machinery.

[4] Nirav Atre, Justine Sherry, Weina Wang, and Daniel S. Berger. Caching with Delayed Hits. In *Proceedings of the 2020 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM)*, SIGCOMM '20, page 495–513, New York, NY, USA, 2020. Association for Computing Machinery.

[5] Benjamin Carleton and Nirav Atre. Delayed Hits in Multi-Level Caches. In *Symposium on Operating Systems Principles (SOSP) (Poster Session)*, 2021.

[6] Erica Chiang, Nirav Atre, Hugo Sadok, Weina Wang, and Justine Sherry. Robust Heuristics: Attacks and Defenses on Job Size Estimation for WSJF Systems. In *Proceedings of the 2022 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM) (Poster Session)*, 2022.