Teaching Statement
Nik Sultana

I have taught in a variety of settings: individual mentoring, small groups, and large groups (over 100 people). When teaching I strive to make connections between the subject matter and real-world examples and recent research. I believe that helps me convey a subject better. My own research serves as a source of such examples for me, and I use it as an engine for teaching particularly when mentoring research assistants or students working on their projects.

Teaching interests. I am enthusiastic about teaching across depth and breadth—ranging from introductory undergraduate courses to advanced, specialized graduate courses and research-level seminars. Having done research both in theory and in systems, I can both be flexible in the subjects that I teach, and also make connections between different subjects.

In particular, I can teach a range of courses on networking including introductory networking, datacenter networking, and programmable networking; systems including operating systems, distributed systems, cloud computing, and system modeling; and programming languages including paradigms, compiler implementation, and static analyses. From my past research, I can also teach aspects of these courses related to security and principles of engineering.

I am keen to build on my experience with flipped-classroom teaching and make more use of online tools. Currently I am leading a project to develop an online teaching and demonstration platform for networking called FDP, designed for both flexibility and ease of use by students.

Mentoring and advising. Altogether I have worked with 23 undergraduate or Masters students in this capacity: 4 undergrad bachelor projects, 5 undergrad research assistants, 12 Masters research assistants, and 2 individual projects for Masters students.

I tend to deeply involve research assistants in my research, and this has resulted in one student being lead author of a paper, three submitting posters, 16 co-authoring papers with me—in the case of four students, more than one paper—and all students contributing to systems that have been released or are being released. Two students won awards for their work.

When working with research assistants, I always use a distributed workflow similar to that used in open-source projects, and students learn how to use source-code revision tools, have their code reviewed, review the code of others, and strive to write clear code, documentation, and adequate tests. This workflow continued operating smoothly when we transitioned to remote work during the COVID-19 pandemic.

Students get good practical foundations in the course of our work, and this helps them regardless of whether they continue working on research or move to industry. In addition to developing their programming skills, students get a better understanding of how shared computing resources are managed and maintained, become more familiar with tools like debuggers, and typically make heavy use of virtualization.

While this extra methodological effort adds some overhead to the research, it also helps us improve the quality of our research outputs and thus their potential impact outside of our project. In two occasions students contributed research-related changes to upstream open-source projects, and the students got to participate in the external code review process and get exposure for their contribution. Virtually all our work is released as open-source. Each project repository’s history shows the students’ contributions, and this can help support the students’ applications to future advisors or employers.

1“there’s so many situations at work, where my team tells me they’re surprised I know best industry practices already, and I always say it’s because of my mentor at grad school!” (former Masters advisee)
Teaching experience. In 2016 I lectured the Prolog course to all second-year computer science undergraduates at Cambridge University, amounting to over 100 students. This was taught in a flipped-classroom approach. The first lecture introduced the course and took place in a lecture hall. Students would then watch prerecorded lectures by Dr Andy Rice in their own time and at their own pace. The subsequent lectures were delivered as lab sessions where I would set out problems for students to solve, and help students with any questions. I also answered questions online, ran the assessed exercises and wrote the exam question together with Dr Alastair Beresford. The course feedback was very positive: 36% of respondents found it exciting and 64% found it interesting.

In 2019 I gave a guest lecture on Denial-of-Service attacks and mitigations as part of CIS331 (70 students), the introductory course on computer security at the University of Pennsylvania. During the lecture I gave several examples of Denial-of-Service (DoS) from the news, described a taxonomy of such attacks and interactively worked through mitigation ideas with the students. I later formulated exam questions related to my lecture.

While a grad student and post-doc at Cambridge University I taught 13 courses as part of Cambridge’s small-group teaching system of “supervisions”. These courses spanned across all three years of the undergraduate curriculum. The courses were: Foundations of computer science, Discrete maths, Logic and proof, Operational semantics, Denotational semantics, Prolog, Concepts in programming languages, Optimizing compilers, Software and interface design, Specification and verification, Compiler construction, Computer networking, and Unix tools.

In a small-group setting, you can work through problems together, hand students the marker and let them drive the thinking at the whiteboard, and guide their interest beyond the syllabus. I found small-group teaching to be very stimulating because it permitted more responsiveness to the curiosity of students and to difficulties they might be encountering.

Teaching philosophy. My teaching experience has provided me with opportunities to reflect on how to teach better. Only when it came to teaching did I realize that course slides are structured very differently from how I structured slides for research seminars. Or how important the first and last 10 minutes of a lecture are.

I learnt the importance of continuously motivating what I am teaching. And how project-based and research-related work can help connect students with externally-validated sources of motivation, helping them see a bigger picture.

Another lesson consisted of the importance of feedback. Only after teaching did I appreciate how different feedback loops can be set up by a teacher. I learnt a lot from seeing how others channeled feedback into better teaching outcomes— for example, through regular in-class quizzes, quick-fire questions, and a well-coordinated group of teaching assistants.

While teaching I came to appreciate that critical thinking should be a side-effect of teaching, regardless of the subject matter. In my own teaching, I try to exercise this by presenting alternative plausible rationales for how a design objective could be met, for example. Since I work in a technical subject, I also try to exercise whole-system thinking to help convey the importance of thinking at different scales and across sub-system boundaries.

Finally, patience is a virtue when teaching too. I recognize that learning can induce a personal maturing of sorts, and that this can take time. Related to this is the importance of allowing students to grow out of their comfort zone, for example while they develop independence, and problem-solving and team-work skills.