I am excited to pass on my knowledge to the next generation of computer scientists. This is one of the main reasons for me to pursue an academic career. In this statement, I discuss my teaching experience, proposals for future teaching, and mentoring experience.

Teaching

My teaching philosophy is based on two effective but often overlooked concepts: **"learning by doing"** and **"the best way to learn is to teach"** (i.e., the Feynman Technique). I have benefited from these two teaching concepts personally as a student and as a teacher. In fact, this is what made me an effective TA at UCLA. I will continue to apply this successful experience in my future teaching.

Past Teaching Experience

As a TA at UCLA, I have mainly taught operating systems and computer architecture. Students often find these two courses challenging; they find the material in these two courses difficult to understand, homework time-consuming, and they still cannot grasp a significant portion of the taught material. I faced this challenge when I started teaching these courses and began to look for ways to address it.

Having taught for several quarters and relating to my own experience, I have found the answer: *hands-on experience*. As a graduate student researcher, I constantly hack operating systems and computer hardware. That is why I found these topics to be generally easily accessible. Therefore, during teaching, I focus on providing students opportunities to "get their hands dirty" so that they are not intimidated by the homework, projects, or exams.

As one example, when I teach operating systems, I need to review how the ext2 file system stores file content with direct and indirect block pointers. Besides a brief presentation, I use an example to ensure students understand how the mechanism works. Specifically, the example starts with a small file that consists of a few blocks. Next, the application appends data to the file, and I ask students how exactly the state in the file system would change. This makes students carefully think about how the mechanism works and interacts with other parts of the ext2 file system. After students have given me a satisfactory answer and I am convinced that most of the students have understood clearly how it works, I would finally ask students to come up with the pseudo-code to handle the write system call in the file system, which provides more hands-on experience. I believe this approach is extremely effective, as reflected in the evaluation feedback:

"*Diyu was an excellent TA. The discussion was well organized and the slides he prepared helped me on every project by giving me an idea of how to start writing code. He also challenged the students during discussion by posing interesting questions that helped me better understand some class concepts.*"

I have also applied the other principle: "the best way to learn is to teach" generally during teaching and particularly during office hours. Specifically, I have helped students learn through the Feynman Technique. For example, whenever students ask me about a homework problem related to, say, the single-cycle implementation of a processor. Instead of directly helping the student with the problem, I would instead challenge the students: "Can you explain to me how does a single-cycle processor work?". Often the student would reply confidently: "Sure!" and then start to explain. I would pretend that I have no background in the material, and raise questions when the explanation is unclear. This forces the student to understand the material better and often makes the student realize that s/he is missing some critical points. Once the student understands the missing points, the homework problem the student asked about earlier would be straightforward. A student commented on this in the evaluation feedback:

"*Diyu is the best there is. Helpful, understanding, thoughtful. We strive to be like Diyu. He provided great help when I had trouble with projects.*"

When I started teaching, things did not go well. I received mediocre teaching evaluation scores and often received harsh comments from the students, such as: "*it was disappointing that it had the most anemic discussions out of all the CS courses*". Sometimes the comments were concise, such as: "*Meh*" or "*Boring*". Such feedback motivated me to constantly reflect on improving my teaching from one quarter to the next. Eventually, this helped me form my teaching philosophy and made me an excellent TA. For the last three years, the overwhelming majority of my teaching evaluation score is with a median of 9 (out of 9) and an average of over 8 (out of 9).

Future Teaching

My expertise is in the general field of computer systems, and specifically, operating systems. I can teach the following undergraduate courses: Operating Systems, Computer Architecture, and Data Structure and Algorithms. I am open to teaching introductory-level courses as well. For graduate-level courses, I can teach Advanced Operating Systems.

I would also be interested in offering a new course: *Data Center Systems* for either undergraduate- or graduate-level. The version for undergraduate students would cover the *internals* of mechanisms widely used in data centers, such as virtualization, VM migration, live update, containers and their orchestration systems, and serverless computing. It would also involve reading several academic papers and a group project to implement one of the taught mechanisms. The version for graduate students would focus on presenting and discussing state-of-the-art system techniques in data centers, such as kernel-bypass IO, resource disaggregated systems, and state machine replication and consensus. Students learn by discussing and critiquing papers and a group project to build new systems for real problems, and, ideally makes new research contributions.

I will continue to apply my teaching philosophy in my future teaching. Specifically, I will share my experience in holding effective office hours using the *Feynman Technique* with my TAs. I will also emphasize *hands-on* experience during teaching. In particular, I will try to make all or a significant portion of the assignments and the exams based on programming instead of a regular Q & A-based exam. For example, one assignment or exam problem would be writing a simplified page fault handler.

Mentoring Experience

As a postdoc at EPFL, I am fortunate to work with many talented junior researchers, including some from underrepresented groups. I believe the key to their success is transparent and honest communication. For example, students often aim for perfection and are afraid to let the mentor know when things do not work. Therefore, during meetings, I explicitly told students that I could make mistakes in research problems. If the approach I propose does not work on the first few attempts, you should let me know, and we can discuss a new solution. This puts students at ease, making them more comfortable to raise objections, which are often correct. Being a graduate student is not easy, and one has to juggle courses, research, teaching, and job search. I often tell the students that if they think the task is unreasonable given other obligations, they should let me know, and we can discuss how to reduce the workload from the non-essential parts of the project. Based on my experience, such kind of communication works well. Students feel the mentor is approachable, has confidence in themselves, and are willing to share their thoughts openly. All these factors are key to a successful project.

I also run a reading group every week. Instead of focusing on the limitations and the design details of the paper, I often present the broad picture of the field and show them how the work advances state of the art. Many students comment that they enjoy reading group a lot and start to understand systems research more from it. Except for technical meetings, I often have informal meetings with students, and we discuss various topics such as managing time, relieving stress, and achieving work-life balance. These issues are also critical to their success, and students really benefit from the discussion. For example, after the intern students finish their projects, they still approach me for this kind of advice.

I am delighted to see that students I work with build up their systems research skills. I co-authored several papers with them. More importantly, many of them are getting interested in doing research and have decided to pursue an academic career.

In the future, I plan to develop an open, collaborative, and friendly research group. Everyone is equal in the group, and we can openly discuss anything. I will also let students collaborate in research projects from the very beginning. Collaboration eases stress, makes students learn from each other, and teaches them the importance of teamwork.