Teaching Statement

Diyu Zhou

My teaching philosophy is based on two **extremely effective, widely known 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, I believe that this is the "secret weapon" that made me a *top-ranked* TA at UCLA and I will continue to apply this successful experience in my future teaching. In the rest of the statement, I discuss my past teaching experience and the proposals for future teaching.

Past Teaching Experience

As a TA at UCLA, I have mainly taught operating systems and computer architecture. Students often find the material in these two courses tremendously difficult to understand, homework or projects extremely timing consuming, and end up getting very little out of the courses. 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 believe that I have found the answer: hands-on experience. Specifically, when I was an undergraduate at Peking University and took operating systems or computer architecture courses, the assigned course projects required me to build an operating system or five-stage pipelined processor from scratch. As a graduate student researcher at UCLA, I modified the Linux kernel and built systems leveraging various hardware features. That is the key reason why I found these topics to be generally easily accessible. Inspired by this, 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, indirect, doubly indirect, and triply indirect block pointers. Besides a brief presentation, I use an example to make sure students understand how the mechanism works. Specifically, the example starts with a small file that consists of a few blocks. Next, the application appends or removes data to or from the file, and I ask students how exactly the state in the file system would change. This forces students to carefully think about how the mechanism works and how it interacts with other parts of the ext2 file system such as block bitmap. After students have given me a satisfactory answer and I'm convinced that most of the students have understood clearly how it works, I would finally ask students to come up with the pseudo code of the write handler 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 the single-cycle processor work?". Often the student would reply confidently: "Sure!" and then starts to explain and I would pretend that I have no background in the material and raise additional questions when the explanation is not clear. This forces the student to gain a better understanding of the material and often makes the students realize 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 for s/he. A student commented 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 how to improve my teaching from one quarter to the next. Eventually, this has 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.

Future Teaching

My expertise is in the general field of computer systems and specifically, operating systems. Additionally, I have extensive experience with the ACM programming contest. For undergraduate courses, I can teach the following courses: Operating Systems, Computer Architecture, and Data Structure and Algorithm. I'm open to teaching introductory-level courses as well. I'm also more than happy to coach or provide my help for the ACM programming contest teams. For graduate-level courses, I can teach Advanced Operating Systems.

If possible, I would also be interested in offering a new course: *(Advanced) Data Center Systems* for either undergraduate- or graduate- level. The undergraduate-level version would be focused on covering the *internals* of mechanisms that are widely used in data centers, such as *virtualization*, *VM migration*, *live update*, *containers and their orchestration systems*, and *serverless computing*. The graduate-level version would be focused on presenting and discussing state-of-the-art system techniques used for data centers such as *Kernel-Bypass IO*, *Resource Disaggregated Systems*, and *State Machine Replication and Consensus*. The undergraduate version would involve reading several academic papers and a group project to implement one of the taught mechanisms. The graduate version would involve reading a large number of papers, and a group project to build a system, that ideally makes new research contributions, based on those systems presented in the assigned papers.

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, if possible, I will try to make all or a significant portion of the assignments and even the exams based on programming, instead of a regular Q&A-based exam. For example, one of the assignment or exam problems would be to write a simplified page fault handler. Inspired by the success of Leetcode, I will leverage a similar platform to ease the burden of grading and allow students to keep refining their code until they succeed on an exam or homework problem.