I believe teaching is an integral part of the journey to advance our collective understanding of the world. A major motivation for me to become a professor is the opportunity to share my knowledge and learn alongside my students. In the following, I discuss my philosophy of teaching courses as well as mentoring students.

Teaching. Richard Feynman put it best: "What I cannot create, I do not understand." My teaching follows the same philosophy of **learning by building**. As a veteran TA of the programming languages course in college, I helped countless students write their first interpreter. Most of them had never used a functional language before, but they come out of the course with their own implementation of a ML-style language with full type inference. Many credit the course as the most important one they took in college, because the core linguistic principles they learn helped them get up to speed with new languages and APIs at work "exponentially faster". The work of myself and other teaching staff contributed to a groundbreaking new textbook, *Programming Languages: Build, Prove, and Compare* (by Norman Ramsey), that is increasingly used in universities across the country.

In graduate school, I have been a TA for the series of courses on database systems. I was part of the teaching staff of every course from Introduction to Databases Systems to an advanced topics class for professional master students. In a particular challenging course, Database Internals, the students build up a fully functioning database, including the buffer manager, query optimizer, and transaction processor. This is one of the largest pieces of software projects throughout our undergraduate curriculum. Not only do the students gain intimate knowledge of the inner workings of a database, but they also become more confident building and maintaining complex software and prepare themselves for the real world.

Outside of the conventional curriculum, I also welcome innovative ways to teach. I organized the first lecture series on History of Computing at UW. The course included a field trip to the Living Computing Museum in Seattle, where students learned to program with punch cards. The history lessons give warmth to the zeros and ones, and lend a human touch to the machines that we work with every day.

Mentoring. My mentoring philosophy follows a simple principle due to Nico Habermann: "Focus on the students, since graduating great students means you'll produce great research, while focusing on the research may or may not produce great students." I have experienced this first-hand at UW, where I have had the fortunate to be mentored by Dan Suciu and Zach Tatlock. Helping the students grow has always been their top priority, and it takes much more than technical prowess: graduate school is a demanding, emotional journey, and kindness and empathy can lift a student out of the abyss to shine like a star.

I have been fortunate to mentor four junior students at UW, and two of them have completed research projects. Jonathan Leang, who contributed to my first publication in graduate school, went on to become a lead instructor of our databases courses, while at the same time working as a software engineer for Snowflake. Another student, Yihong Zhang, won 1st place at the PLDI student research competition, and lead a paper published at POPL (a top PL conference). Some of the most memorable days of my graduate school career were spent optimizing programs and tweaking prose with these students, and I look forward to many more years of that as a professor.