1. Overview

Teaching is much more than an opportunity to engage with students on complex curriculum. As teachers, we can inspire students to be independent, critical thinkers and scholars, all while having a memorable experience. My goal is to be such a teacher.

I have been actively mentoring and teaching at 2 institutions: At NJIT, I have taught or am scheduled to teach: Programming for Computer Graphics, Information Design Techniques, and AI for Artificial Worlds. At UCLA, I have taught 4 different computer science classes for over 12 terms, which include: Software Engineering, Programming Languages, Software Construction, and a CS Freshman Seminar.

Based on my experiences, I can teach the following undergraduate classes, for example:

- 2102. Introduction to Software Engineering
- 3150. C++ Essentials
- 3200. Mobile Application Development
- 3802. Numerical Methods in Scientific Computation
- 4703. Principles of Computer Graphics
- 4705. Artificial Intelligence
- 4820. Introduction to Machine Learning
- 5819. Introduction to Machine Learning
- 5820. Machine Learning
- 6514. Computational Topology

I am also well prepared to teach both beginner and advanced level courses in my research focus. Specifically, I believe that my background in topics such as **real-time optimization**, **simulation**, **game technology**, and **computer vision**, would allow me to bring industry sought-after skills to students. Not only can these offerings enhance your department, but also represent opportunities for interdisciplinary exploration (e.g., machine learning, robotics, human-computer interaction, as well as other empirical sciences).

2. Teaching Philosophy

My approach to learning is embodied in the adage that you should never tell students something they can tell you. I construct my classes around this philosophy, by making them interactive, filled with examples and thought-out questions about each learning point. For example, while teaching OCaml, I had students discuss and solve pattern matching programming challenges with an increasing complexity level. I do not provide a solution to these problems. This serves two purposes, firstly to check the clarity of the ideas presented, and secondly to ask about the "adjacent other", that is, what are the next possible steps and unknowns. From my experience, asking students to provide answers to lots of questions helps to keep them engaged, and combats waning attention spans that often result from continuous lecturing. Additionally, such interactivity should also continue outside of the traditional lecture hall, by using online venues such as *Discord*, where students can discuss class topics with minimal staff supervision.

I strive to keep my class engaged by employing different approaches. First, I motivate my students by explaining that it is better they try to understand how to solve a problem in class, since it is likely they will encounter a similar problem in a professional setting such as a job interview, and that would likely not be an educational and forgiving experience. Second, I use catch phrases to emphasize and differentiate concepts. For example, I use the term "muscle memory" to signify the importance of answering a particular question on a well versed domain immediately and with clarity. In another example, "OCaml land" or "Java world" differentiates between two completely different programming paradigms, such that students should use a radically different way of thinking when approaching problems. Third, I structure my classes so any new topic introduced is combined with questions that check student understanding.

Finally, teaching is a two-way street. As a teacher, you consistently receive questions and feedback from students. Learning from this feedback allows you to become a better teacher. Personally, I learned valuable lessons from listening to students about the kinds of feedback they wanted, and adapting my behavior accordingly. Additionally, this active learning process encourages curiosity and the thirst for discovery. As students learn (by the instructor's example) that asking the right questions can lead to inspiration, they become more active participants in their own education and more mature as scientists as well.

3. Student Mentoring

As faculty, I have collaborated, taught and mentored affiliated students. I worked closely with my mentees and collaborators in all stages of the research process—idea brainstorming, related work researching, coding for simulations and experiments, and technical paper writing. The result is evident by my publications, which were coauthored with lab members and collaborators, including undergraduate students.

I enjoy being a research mentor and project leader, and I believe that such a role is essential in building an outstanding research group. I shared my research experience, and provided them with career advice, taking into account their own backgrounds and goals. I have also learned to tailor my mentoring style to bring out the best in students according to each student's unique personality and working style.

In my research group, I also encourage students to work in pairs or groups. In groups, students are able to learn from each other, and are usually more efficient in problem solving, since two minds are actively working on the same problem, which provides ample opportunities for brainstorming, and an extra set of eyes for catching possible errors. Working in a group also helps students develop teamwork habits and skills, which, I believe, will be essential and beneficial to their future success.