I bring nearly a decade of teaching experience to a computer science classroom. For six years, I taught mathematics and computer science at the K-12 level, specializing in 6th grade Pre-Algebra, 9th grade Algebra II, and 12th grade AP Computer Science A. During my time at Georgia Tech, I served as a teaching assistant for a total of 8 semesters in a variety of topics: Graduate Introduction to Operating Systems (CS6200, three semesters), High-Performance Computer Architecture (CS6290, three semesters), and Undergraduate Introduction to Artificial Intelligence (CS3600, two semesters). As a teaching assistant, I tutored students 1-on-1 and in small groups, designed assessments, ran small group activities, and graded student work. I also completed my Tech to Teaching Capstone by serving as Instructor of Record for CS3600 Introduction to Artificial Intelligence, creating my own curriculum and publishing my activities in EAAI'26. I am a MathStreamer for Carnegie Learning, creating engaging instructional videos for K12 mathematics lessons. My research concentration in AI for Education was initially sparked by my love for teaching, but as I explore how representation can be used to personalize the student experience in technology, I find new ways to express and engage my students in the classroom.

For lifelong success as a computer scientist or software engineer, today's students will need to learn new languages on a fly, have a solid intuition for how computer algorithms work, and have strong independent debugging skills in order to design and build any piece of software. My pedagogy is guided by constructionism, the theory that learning happens by doing. I believe that the most long-term learning comes from discovering concepts for yourself. As a computer science instructor, my primary goal is to give students the experiences and opportunities they need to discover course concepts. I never want to simply tell my students something they could experience for themselves. I want to create opportunities for my students to try, experiment, fail, succeed, and learn. As I plan my instruction, I'm considering how to chunk the content, how to give my students opportunities to practice, when to insert formative assessment, how the students are going to interact with each other and with me, and how I will be measuring the success of my lesson. In this way, designing lessons for 6th graders is not so different from designing lessons for undergraduates, and I can apply my experience in the K-12 classroom to higher education. In both environments, I try to build a classroom where students are confident in their ability to learn, where the learning activities help my students gain an intuition for how computers make decisions and teach them expert debugging skills.

For students to persist in a subject, they need to see themselves as capable of succeeding in it (Steele, 1997, A Threat in the Air). This self-identification comes when students believe they can succeed and gain confidence in their abilities, making them more resilient to setbacks and more likely to continue learning. Especially in subjects like computer science and artificial intelligence, students can feel apprehensive about learning a highly technical subject that is often preceded by complicated mathematics. My goal is to create a safe, supportive environment where students feel free to try new things without fear of failure, and a space where they can see themselves having a future in the field. To achieve this, I design courses that provide accessible entry points to technical material, applying Universal Design for Learning principles and eliminating hidden curriculum wherever possible. I use physical analogies and games to make technical content easy and intuitive, before building a more precise, technical understanding. I try to leverage my student's current knowledge and expertise to build a deeper understanding of challenging content. I aim to give students early 'wins' so they see success is attainable, even if they initially do not see themselves as a computer person. My students have appreciated this approach, saying "*I truly believe she is the best professor ever. I've taken this class before [with a] different professor and the knowledge felt so unattainable, and I just couldn't follow. I thought it was me and that I was stupid, but here I am taking it again and it's all just so simple*". Another student shared, "*She explains concepts in a way a beginner could understand, so I never feel like I'm left behind*", and a third reflected, "*I really appreciate how she does around to everyone and makes sure everyone understands the concept individually as well, so no one feels left behind before class is over*." Outside observers remarked that "*your passion for the material and the relationship you have with your students is the most effective aspect of your teaching*". These comments reinforce my belief that classroom climate and individualized attention matter deeply, shaping how students see themselves and their potential in the discipline.

What I want students to take away from my classes is an intuitive understanding of why things work the way they do. With this intuition, students can more easily reconstruct the technical and mathematical details both during the course and later when applying the material in their career. When intuition comes first, the technical details become more meaningful. Most artificial intelligence methods are based on human intelligence, and students are already experts in their own decision-making. I use physical manipulatives, metaphors, and games to put students in a situation where they need to make an optimal decision under stochasticity or uncertainty, such as card or dice games. Through these activities, the world dynamics become clear and students can reason concretely, even when outcomes are uncertain. When we follow each activity up with a mathematical formalization of how AI would represent and process the problem, students can see how and why we formalize phenomena in particular ways, instead of seeing arbitrary definitions before grasping the big picture. Students have expressed how this focus on intuition helps them bridge the gap between theory and practice. One shared "*I really enjoy the activities we do in class. I think they've helped solidify my understanding of a lot of concepts that seemed very abstract. Tracing through small examples of these* *algorithms running on the board makes it easier to visualize and extrapolate them*." Another noted "*The simulated demonstration as well as the hands-on activity made the concepts less abstract, allowing me to learn them tangibly in a way.*" A third student said that "*the physical activities were really helpful to me since it let me physically model the actions an agent would take and learn about how it would practically work over just learning theory*". Building on my student's ability to reason about an activity makes the transition to abstract theory and technical details easier, and they now have a lived experience to tap into when using each concept in the future.

Independent problem-solving is a skill that students will have to use in courses outside mine, and for the rest of their technical careers. As an instructor, it can be so easy to pinpoint what a student is doing wrong from common bug rules, and the fastest way to solve student problems is to tell them what is wrong in their work. But having a student inspect their own work more thoroughly is a more valuable learning experience. Instead of telling students what is wrong, giving them hints about what to inspect, how to use a debugger, and encouraging them to think about what data structure they should investigate at what stages takes longer to solve their problems, but students will internalize these informal lessons. Through debugging together, students learn how to self-verify the correctness of their work, and how to fix their own problems independently. One student said "*She had super unique but effective ideas for debugging. She was genuinely interested in helping you solve your issue and not just giving you a solution to try. I have never met a TA this invested in the students and this good at what they do!*" This kind of feedback reassures me that students are not only learning to fix immediate problems but are also developing the mindset and skills to diagnose and solve their own problems independently.

While course content is important, I also want to prepare my students for a future of lifelong learning. My classroom prioritizes student confidence and well-being. I want to give my students the tools they need for success beyond my course, like computing intuition and independent debugging skills. I would enjoy teaching introductory classes like **Data Structures**, **Intro to AI/ML**, or **Algorithms**. I love being a student's first introduction to a subject, so I can show them how interesting this content can be and show them how they can see themselves as a computer scientist, an AI engineer, or an application developer. I would enjoy teaching cross-over classes like **Math for AI** or **Educational Technology**. Finding overlap between two topics makes content more meaningful and showing students how math is found in CS or how learning theories are evident in Ed Tech can help them see there's more to pursing a subject than pure CS theory. I would also enjoy going deep in a specific subject, like **High Performance Computing Algorithms** (applying parallelism to solve problems efficiently across multiple cores) or **Computational Models of Learning** (Bayesian Knowledge Tracing, Performance Factors Analysis, and AI that

Commented [RJM1]: separate out from the what classes i would teach models human cognition). You'll find that I am an experienced educator with nearly a decade of teaching experience, enthusiastic about making higher education accessible.

Commented [RJM2]: Talk about individual mentorship and your independent studies. Indicators of success of mentorship (authors on papers, any awards or jobs of note)

how many students you've mentored. undergrad and graduate.