JONATHAN CUI

Programming Languages

Python - 5 years - proficient
Rust - 3 years - intermediate
Nix - 1 year - intermediate
CUDA - 3 years - familiar
React - 4 years - proficient

EDUCATION

Double-major in Computer Science and Mathematics — 2022–2024
B.S., Penn State University, 4.0 GPA

Computer Science, Minor in Mathematics — 2024–2026
B.S., UC San Diego, 3.97 GPA

EXPERIENCE

Research Intern (Unpaid), Microsoft Research Asia — 7/2019–2/2020

- Machine learning research on computer vision and medical applications.
- Two papers published as the second author.
  - One paper is published as a *Spotlight Paper* at ICLR 2021, which I presented at the conference.
- Responsibilities included experimentation, data collection, manuscript drafting and revision, and demo implementation.

Software Engineering Intern, Starlink (SpaceX) — 6/2025–9/2025

- Built AI tool integrations supporting thousands of global support agents with 50k+ daily API calls.
- Contributed to a core AI SDK re-architecture, enabling a 3× increase in internal adoption and traffic.
- Maintained external facing support pages in Next.JS serving millions of users.

RESEARCH

https://scholar.google.com/citations?user=oRtbHw4AAAAJ&hl=en

- 4 peer-reviewed publications with 400+ citations.
- Presented Spotlight Paper at ICLR 2021.

PROJECTS

TasteMate, AI Recipe App — https://tastemate.pro/

- TasteMate is an AI recipe app tailored to people with medical conditions and diet restrictions.
- The app is written in Flutter and the backend in Rust. The website was crafted with Next.JS.
- Configured server and services declaratively with Nix flakes, deployed on a Proxmox VM.

Curry–Howard Proof Verifier — https://math.joncui.sh/

- A dependent-type proof verifier for second-order logic with Rust-style generics.
- The engine is written in Rust, with a handcrafted recursive descent parser in chumsky.
- The website serves demos on real-time provisioned micro VMs with Firecracker on a Rust server.

Distributed GPU Training on Spot Instances

- Cut env setup time from ~6 hours to <20 minutes by scripting reproducible Docker-based pipelines.
- Built distributed experiment dashboards in W&B for cross-node training metrics.
- Optimized PyTorch multiprocessing dataloaders to reduce GPU idle time during gradient synchronization, improving efficiency across 4–8 GPUs.