Mario Lucido - **Availability:** Full-time start**, Summer/Fall 2026**

MarioLucido94@gmail.com | (707) 301-0025
Website: mariofoint.github.io | ***LinkedIn***: www.linkedin.com/in/mario-lucido-b556a628a

Education

Sonoma State University, Rohnert Park, CA
Bachelor of Science in Computer Science (Minor: Philosophy)
GPA: 3.89 | **Dean's List** (Fall 2023 - Spring 2025)
Expected Graduation: May 2026
Relevant Coursework: **Computer Architecture, Quantum Computing, Data Structures, Operating Systems, Algorithm Analysis, Programming Languages**

Technical Experience

- **Languages:** Python, C++, SQL, Java
- **ML/AI:** PyTorch, ONNX, TensorRT, CUDA, ONNX Runtime
- **Systems & Infra:** Inference Optimization, Multi-threading, GPU Acceleration, Performance Benchmarking, Kernel Profiling, Amdahl's Law
- **Data Pipelines:** NumPy, Pandas, Matplotlib, scalable batch processing
- **Tools:** Git, VS Code, Chrome DevTools
- **CS Fundamentals:** Algorithm design, complexity analysis, low-level memory optimization

Experiences & Projects

Technology Intern OurCo*, Spring 2025-Present*

- Led QA testing and bug triage for new cross-platform features (Web, iOS, Android) in an Agile environment using GitHub and Chrome DevTools.
- Retested and documented issues with full reproducibility; authored internal onboarding documentation for incoming interns.

GPU Inference Benchmarking – Loan Risk Predictor*, Summer 2025*
https://github.com/mariofoint/loan-risk-gpu-vs-cpu

- Trained a PyTorch-based classifier on real financial data (~2M records) and exported to ONNX for deployment-style benchmarking; modeling similar to ad prediction pipelines.
- Benchmarked inference latency using ONNX Runtime on CPU, CUDA, and TensorRT backends.
- Measured scaling behavior from batch size 1 → 10,000 using ONNX Runtime (CPU, CUDA, TensorRT) and visualized latency trends with Matplotlib.

CUDA Julia Set Fractal Renderer*, Spring 2025*
https://github.com/mariofoint/CS-351-computer-architecture/tree/main/Project-4

- Built a CUDA-accelerated Julia Set renderer and benchmarked against multithreaded C++ implementations.
- Analyzed GPU kernel launch overhead, memory transfers, and vector scaling behavior for large data sizes.