

ZIANG (ANDREW) TAN

Seattle, WA | ziantg2@uw.edu | 206-601-7855

linkedin.com/in/ziantg-tan | github.com/ziantg2 | ziantgan.com | Availability: June–September 2027

EDUCATION

University of Washington | Seattle, WA

B.S. Electrical & Computer Engineering | B.S. Applied Mathematics: Data Science | Expected Dec 2027

B.S./M.S. Electrical & Computer Engineering, M.S. | Expected June 2028

Relevant Coursework: Data Structures & Algorithms, Machine Learning, Embedded Systems, Computer Architecture

TECHNICAL SKILLS

Programming: Python, C++, Java, C, SQL, Bash

Robotics / Systems: Linux, Git, TCP/IP, SSH, FreeRTOS, CUDA, TensorRT, NVIDIA Jetson Orin Nano, Raspberry Pi

ML/Vision: PyTorch, OpenCV, Computer Vision, SigLIP2, FAISS, Multimodal Retrieval, Vision-Language Embeddings

Backend & Cloud: RESTful APIs, FastAPI, AWS (EC2), Docker, Kubernetes, Kafka, Redis, CI/CD, Agile/Scrum

RESEARCH EXPERIENCE

ARCLab, University of Washington | Seattle, WA

Undergraduate Researcher — Tiny-Aircraft Detection for Vision-Based UAV Detect-and-Avoid | February 2026 – Present

- Built a reproducible **Python** and **PyTorch** experiment pipeline across 72K synthetic images and 1,799 real images for repeatable model evaluation work.
- Layered tiled inference, cached metrics, validation-only threshold selection, and frozen-test safeguards into the pipeline evaluation design chosen.
- Designed a high-resolution **YOLO26n** tiny-object detector using **P2** feature fusion and custom refinement modules for small-target detection systems.
- Lifted sub-20-pixel recall from 61.1% to 71.3% through the refined detector architecture while simultaneously reducing false positive detections.
- Built a failure-driven synthetic-to-real data-selection pipeline prioritizing model misses, lifting frozen-test recall by ~2.1x under a fixed budget.

SELECTED PROJECTS

EdgeML Robotic Perception & Autonomous Sorting | Python, PyTorch, OpenCV, TensorRT, Jetson Orin Nano, TCP/IP, Raspberry Pi

- Architected an end-to-end robotic perception and control stack across an **NVIDIA Jetson Orin Nano** and **Raspberry Pi** for autonomous pick-and-place.
- Separated vision inference from actuator control, sending detected targets over **TCP/IP** to a mechArm for real-time autonomous grasping operations.
- Optimized **YOLO** inference on Jetson with FP16 **TensorRT**, raising throughput from 20.8 to 27.6 FPS against a comparable 52.4 FPS **OpenCV** baseline.
- Implemented failure-handling and recovery paths for rescanning, verification, and re-localization steps across the entire robotic perception stack.
- Validated 10 physical scenarios on a leakage-audited dataset of 193 video clips, reaching 18/18 successful grasps and 15/18 verified placements.

Multimodal Retrieval & Recommendation System | Python, PyTorch, SQL, SQLite, SigLIP2, FAISS, Information Retrieval

- Built a **Python** and **SQL** retrieval and ranking pipeline over 330K interaction events spanning 15,000 distinct users and 600 catalog products.
- Layered normalized **SQLite** storage, indexed queries, and automated leakage checks into the pipeline, reaching 0.266 Recall@10 on a 20-test suite.
- Built a separate multimodal product-retrieval benchmark over 1,000 public Amazon Berkeley Objects listings using pretrained **SigLIP2** image embeddings.
- Reached 89.3% Recall@10 and a 0.781 MRR score on 523 held-out image views using **FAISS** for fast approximate nearest-neighbor retrieval search.

Course Scheduling API Service | Python, FastAPI, RESTful API, AWS, Docker, CI/CD

- Formed a **RESTful API** service in **FastAPI** for a course-scheduling app, exposing endpoints for enrollment, search, and schedule conflict detection.
- Deployed the service on **AWS EC2** behind a **Docker** container, keeping the runtime environment consistent across development and production servers.
- Wired a **GitHub Actions CI/CD** pipeline running integration tests on every single pull request before merging changes to the shared main branch.
- Cut manual deployment steps from a 30-minute checklist down to a single automated push, reducing release friction for the whole small dev team.

Event-Driven Notification Pipeline | Python, Kafka, Redis, Kubernetes

- Devised an event-driven notification pipeline streaming user activity events through **Kafka** to a set of independent downstream consumer services.
- Cached recent notification state in **Redis**, preventing duplicate alerts from firing when the same event was processed more than once accidentally.
- Deployed the consumer services on **Kubernetes**, letting the cluster automatically scale worker replicas up during periods of unusually heavy traffic.
- Ran the project in two-week **Agile** sprints with a small 3-person team, reprioritizing consumer coverage based on weekly retrospective team feedback.