Ilyaas Kapadia

kapadiiy@mail.uc.edu · (513) 257-8525 · ilya.as · github.com/IlyaasK · linkedin.com/in/ilyaask

Available for Spring 2027 Co-op

Education & Skills

University of Cincinnati · Bachelor of Science + Master of Engineering, Computer Science May 2029
Coursework: Discrete Math, Computer Systems, Data Structures, Stanford's CS231n (in progress)
Languages & Systems: Python, C, C++, CUDA, Rust, Go, Bash, SQL · PyTorch, PufferLib, MuJoCo Warp, Docker, RL, Evals

Work Experience

Software Engineer Intern · Kernel (YC S25) May – Aug 2026

- Slashed CI test pipeline runtimes from **15m to 57s** (**$23K/yr saved**)
  - Fixed Go compiler cache invalidation by aligning toolchain paths under non-login shells (`bash -c`)
  - Replaced full-database `TRUNCATE` with connection-scoped MySQL transaction rollbacks (`BEGIN … ROLLBACK`)
  - Split 424 pure-logic tests (0.67s) from 117 DB tests (5s), cutting dashboard suite from **8m09s → 4.46s**
- Architected and released Kernel's official **Terraform Provider (`v0.1.0`)** from scratch
  - Built resources with HCL validators
    - Ensured drift-free state normalization for Chrome-policy JSON
    - Ensured zero float-precision mismatches against the Kernel API
  - Implemented canonical import ID resolvers so infrastructure can be adopted without destroy/recreate
  - Built an acceptance harness with durable cloud fixtures and automated state verification
- Stripped legacy headers from Kernel images, reducing image size by **300 MB+** and cold build time by **50s**
  - Replaced external network e2e tests with an in-process hermetic `httptest.Server`
- Added `AbortSignal.timeout` deadlines to browser automation across setup and CDP handshakes
  - Prevented orphaned browser instances from leaking container memory on client disconnects
- Implemented end-to-end API key generation across the Go SDK, MCP server, CLI, and React dashboard
  - Enforced project-scoped authorization, blocking privilege escalation and self-deletion

Projects

Mario Eval · ilya.as/mario-eval Nov 2026

- Built an open-ended 24h RLVR optimization eval on *Super Mario 64* after NES Eval topped at **0.986**
  - Anchored honest baselines via a 17-patch ablation ladder (**1.00×** IDO, **1.34×** `-O2`, **1.66×** Kaze)
- Resolved ambiguous ground truth and false negatives via **Player Equivalence** (**25** routes, 19 held-out)
  - Matched actors, **31** render variants, and log-mel audio (**0/11** false negatives; **5/5** cheats caught)
  - Keyed inputs to logical frames to stop speedup desync; mash routes caught 1-ULP trig drift
- Closed gaps between measured and intended N64 behavior via C++ `ares` profiling and red-teaming
  - Fixed **0-cycle** uncached-RAM/idle loopholes, **+3.5%** `#ifndef PLAY` splits, and `envaudit` `NaN` bugs
- Turned 24h multi-harness agent runs into zero-transcript research signals and fault attribution
  - Separated runs (**1.05×–1.64×**, **98%** Kaze); auto-attributed **+0.36×** overfit, `FPE` traps, and cache UB

NES Eval · ilya.as/nes-eval Oct – Nov 2026

- Built a 24h lockstep WASM eval (extending GBA Eval; **29** cases, **44,980** frames, **0.06–0.986**)
- Eliminated grader noise from ambiguous NTSC colors, random boot state, and host FP drift
  - Graded 9-bit PPU palette indices instead of RGB and pinned boot state so valid palettes match
  - Enforced byte-identical CI determinism (**146** files) via WASM `NaN` canonicalization and `BTreeMap`
- Audited a **0.362**-point gap between measured score and skill (**0.58 → 0.94** on untuned `nes-core`)
  - Fixed an unstated first-call no-op found via oracle probing; stripped `RustyNES` ROM provenance
- Built mutation auditing (`envaudit`) and v2 splits (held-out ROMs, seeded mash, CPU/RAM fuel)
  - Closed unbilled `emu_set_keys()` **1.00** fuel bypasses and log-mel CMN scoring muted APU at **0.961**

Reinforcement Learning Simulation Engineering · PyTorch, PufferLib, C, CUDA, MuJoCo Warp Sep – Nov 2026

- **Slither.io Multi-Agent Self-Play · ilya.as/slither**
  - Engineered a CUDA-vectorized simulation with PufferLib 5, training a PyTorch policy at **5M steps/sec**
  - Trained a policy that achieves a **10,000+ score** in live online play after 10B steps (60 minutes)
- **Single-Propeller Drone RL Control Policy · ilya.as/monocopter**
  - Built a flight aerodynamics simulator in PufferLib training at **5M steps/sec**, learning to hover in under 10 mins
- **Brotato RL Environment · ilya.as/brotato**
  - Replaced the Godot engine and 454 GDScripts with pure C/CUDA running at **2M steps/sec**
  - Authored a custom in-game mod for live policy playback, reaching **Wave 4/20** in real gameplay
  - Cut per-env state to 1.3 MiB and removed host staging via typed index widths, bitmasks, and ring pools
- **MuJoCo 2-DoF Self-Rising Robot · ilya.as/self-rising-robot**
  - Built a 2-DoF bipedal contact sim running at **3M steps/sec** with PufferLib 5 on RTX 5070
  - Achieved 6x MJWarp throughput by removing Python and GPU-to-host communication and specialized physics

- Policies transfer to MuJoCo CPU and achieve sim-to-real transfer