Aidan Ewart

aidanprattewart 'at' gmail 'dot' com
github.com/Baidicoot
linkedin.com/in/aidanewart

SUMMARY

I am an AI safety researcher working on improving red-teaming and safety quantification in language models. I have authored a number of papers in LLM adversarial robustness and interpretability. I am an experienced programmer (8+ years) and am proficient at a number of programming languages.

EDUCATION

University of Bristol Sept 2022 - July 2025
Undergraduate Mathematics (Current) *1st in all graded classes*

EXPERIENCE

Haize Labs July 2024 - Oct 2024
Research Intern *New York, NY*

- Early stage startup, interned when they had <5 employees
- Built up their automated red-teaming infrastructure, developing novel automated attacks
- Contributed to Haize's contract work with Anthropic and red-teaming of the OpenAI o1 models

MATS 5.0 Jan 2024 - Mar 2024
Scholar *Berkeley, CA*

- Developed a post-training method (T-LAT) for unlearning and adversarial robustness in LLMs
- T-LAT was a Pareto improvement over prior robustness methods against automated attacks
- Demonstrated applications in removing backdoors inserted via data poisoning
- Implemented and ran distributed versions of T-LAT

SELECT PUBLICATIONS

Sparse Autoencoders Find Highly Interpretable Features in Language Models ICLR Conference Paper
H. Cunningham, **A. Ewart***, L. Riggs*, R. Huben, L. Sharkey*

- Conference paper at ICLR 2024
- Workshop paper at ATTRIB @ NeurIPS 2023

Targeted Latent Adversarial Training Improves Robustness to Persistent Harmful Behaviors in LLMs NeurIPS Workshop Paper
A. Ewart, A. Sheshadri*, P. Guo, A. Lynch, C. Wu, V. Hebbar, H. Sleight, A. Cooper Stickland, E. Perez, D. Hadfield-Menell, S. Casper*

- Upcoming Workshop Paper at SoLaR @ NeurIPS 2024

Eight Methods to Evaluate Robust Unlearning in LLMs Preprint
A. Lynch, P. Guo*, **A. Ewart***, S. Casper, D. Hadfield-Menell*

SELECT PROJECTS

Functional Programming Language Compiler Source on GitHub
Haskell, x86 Assembly, C

- Implemented a compiler for a Lisp-like high-level programming language
- Frontend includes Hindley-Milner typechecking and inference, a module/imports system, compilation with continuations
- Backend includes program optimisation, register allocation, compilation to x86 assembly and C

Proof Assistant Source on GitHub
Lua, Haskell

- Implemented a theorem-proving DSL for Lua
- Proof assistant uses a Martin-Löf style type system complete with type inference via unification
- Includes a customisable notation system in the style of Coq