MIKE HE

https://ad1024.github.io

dh63@cs.washington.edu
(206) · 887 · 8588

Education

University of Washington, Seattle *Sept. 2018—Est. Jun. 2022*
B.S. in Computer Science

- Major GPA: 3.79 / 4.00
- Fields of Study: Programming Languages & Formal Verification & Compilers & MLSys
- Honors: Dean's List (College of Art & Science) 2018–Now; CRA Outstanding Undergraduate Researcher Award nominee of the Allen School (2021)

Publications

1. Marisa Kirisame*, Steven Lyubomirsky*, Altan Haan*, Jennifer Brennan, **Mike He**, Jared Roesch, Tianqi Chen, and Zachary Tatlock. Dynamic tensor rematerialization. In *International Conference on Learning Representations (ICLR'21)*, 2021 (*: Equal Contribution)
2. **(Under review at PLDI '22)** Bo-Yuan Huang, Steven Lyubomirsky, Yi Li, **Mike He**, Thierry Tambe, Gus Henry Smith, Akash Gaonkar, Vishal Canumalla, Gu-Yeon Wei, Aarti Gupta, Sharad Malik, and Zachary Tatlock. Speicalized accelerators and compiler flows: Replacing accelerator apis with a formal software/hardware interface, 2021

Workshop Papers

1. Bo-Yuan Huang*, Steven Lyubomirsky*, Thierry Tambe*, Yi Li, **Mike He**, Gus Smith, Gu-Yeon Wei, Aarti Gupta, Sharad Malik, and Zachary Tatlock. From dsls to accelerator-rich platform implementations: Addressing the mapping gap. In *Workshop on Languages, Tools, and Techniques for Accelerator Design (LATTE'21)*, 2021 (*: Equal Contribution)

Experience

3LA, LATTE '21 June. 2020—Now
Research Assistant @ PLSE *Seattle, WA*

- 3LA proposes an end-to-end compilation flow that provides **flexible** and **verifiable** compiler support for custom Deep Learning (**DL**) accelerators. 3LA has a builtin implementation agnostic pattern matching algorithm that is capable of find accelerator supported workloads in DL models leveraging the power of Equality Saturation. Moreover, 3LA addresses the mapping gap between DL models represented in high-level domain-specific languages (DSLs) and specialized accelerators using Instruction-level Abstraction (**ILA**) as the software-hardware interface.
- Co-authored a Workshop paper and a conference paper (under review at PLDI)
- Designed and led the implementation of Flexible Matching algorithm to match supported workloads
- **Talks & Presentations**:

1. *From DSLs to Accelerator-rich Platform: Addressing the Mapping Gap*, Sept. 2021 at Intel (presented jointly with Steven Lyubomirsky)
2. *Correct & Flexible Compiler Support for Custom Accelerators*, Sept. 2021 at SRC ADA Center

Dynamic Tensor Rematerialization, ICLR '21 Oct. 2019—Aug. 2021
Research Assistant @ PLSE *Seattle, WA*

- Dynamic Tensor Rematerialization (**DTR**) is a greedy gradient checkpointing algorithm. DTR **enables** training Deep Learning models on memory-constrained devices. Unlike previous approaches, DTR does not need any information of the DL model architectures ahead-of-time; instead it saves memory by evicting and recomputing tensors **on the fly**, i.e. trading time for memory, which further exploit opportunities of using gradient checkpointing on DL trainings. DTR is comparably efficient as previous approaches: it requires only $\mathcal{O}(N)$ more forward computations when training a N-layer linear feed-forward neural network with an $\Omega(\sqrt{N})$ memory budget.
- Co-authored a paper accepted at ICLR'21 Spotlight track
- Developed and improved the evaluation infrastructure of the prototype implementation in PyTorch

Paul G. Allen School, University of Washington Mar. 2021—June. 2021
Teaching Assistant *Seattle, WA*

- Worked as a TA for **Principles of Programming Languages** (CSE 505)
- Helped re-designing CSE 505 and developing course materials for various topics about PL and formal verification (**Hoare Logic**, **Lambda Calculus** and **System F**, etc.) in **Coq**.

Service

→ **MICRO '21**, Artifact Evaluation

Projects

veripy

- An easy-to-use auto-active program verification library for Python programs written in **Python**.
- The library is shallowly embedded in Python and the interface is implemented as **decorators**. It compiles annotated Python functions to **SMT** formulae and calls **SMT solver** to check whether it matches the given specification and gives a counter-example input when it violates any constraint.

dtlc

- Implemented **dependently-typed** lambda calculus in Martin-Löf style intuitionistic type theory.
- Written in **OCaml**, **dtlc** has a language frontend **Lexer & Parser** implemented using **Menhir**. Core language supports **type unification** with **metavariables** which makes the type inference stronger.
- Implemented eliminators for **naturals**, **identity type**, **union type**, etc.

Sager

- Course project of CSE 507 - Computer-aided Reasoning for Software
- A demonic data structure synthesizer written in ROSETTE, a **solver-aided** language based on **Racket**.
- **Sager** is able to exploit algorithm bottleneck by performing **symbolic exeuction** over the whole algorithm and using **SMT solver** to synthesize a sample data structure the algorithm works on that pushes the algorithm to its worst-case performance.