Tran Ma

tranma | tranma | ma.ngoc.tran@gmail.com | +61425896023

About

I am a software engineer with experience building data-centric applications, including computer vision models. I have strong research skills, and a passion for machine learning.

Work Experience

Ghost Autonomy Mountain View, CA, USA and Sydney, AU

Machine Learning System Engineer April 2019 - March 2024

I worked on Ghost Train, our internal *training and deployment* system for vision models. Ghost Train was a key part in the production driving pipeline, which achieved thousands of autonomous miles on mobile hardware.

- I wrote a *library for building models* as Scala programs, analogous to PyTorch models in Python. Most of Ghost software was in the Scala/Java ecosystem, and this enabled engineers to seamlessly integrate with other parts of the platform such as the camera sensor stack and performance testing.
- I implemented several *array operations* and their matching gradient code for CPU and GPU (with OpenCL). These operators ran in both training and inference in the car. Having the same implementation for all cases improved confidence in the model and reduced maintenance overhead.
- I tested the evaluation and back-propagation engine used for training. I used fuzz testing techniques to find errors where shared model structures broke the algorithm.
- I wrote the *compiler to produce inference code* for models trained with our system. The produced inference code contained only optimized operators and static buffers, removing any overhead not necessary for running the model in the car.
- I *accelerated the inference code* using graph rewrite rules and operator fusion, which reduced the required GPU memory bandwidth. These techniques were the first major performance gains, bringing our perception models to a realistic runtime profile for development.
- I *specialized operator implementations* to take full advantage of the compute and memory model on our target GPU, the Qualcomm Adreno 630. I used empirical experiments and benchmarks to discover optimal input partitions and memory locations, producing convolution implementations specialized to the shapes of input data. These optimizations were key to bringing our production models to real-time performance in the car (30fps).
- I *optimized array layouts* in the compiled model to weigh the cost of shuffling data order with better operator performance when they are shuffled. This was necessary to get our biggest models to run in the car.
- I *quantized* our models to use low-precision numeric representation. This took better advantage of the memory model on our GPU without significantly reducing model accuracy.
- I wrote an exporter to produce ONNX description files from our models. Using ONNX allowed us to efficiently deploy trained model weights, as well as interact with other tools, such as Netron for visualization.
- As the deep learning ecosystem matured, we began to migrate from Ghost Train to PyTorch for training,

and Qualcomm Neural SDK for deployment to the car. We maintained Ghost Train support for models which the Qualcomm toolchain could not support. I debugged problems with the toolchain and identified pitfall areas, such as low-precision operators and specific shapes of input data.

I worked on *lane detection* in the model engineering team, building a lane model in birds-eye view (BEV).

- I implemented known BEV models for lane detection and evaluated them on our data, identifying problems due to differences in data distribution with regard to camera pitch and road geometry.
- To improve our BEV model, I researched ways to represent complex lane geometries such as splits/merges and intersections in BEV.
- To produce ground-truth to train our BEV model, I built a lane identification algorithm based on a probabilistic method, which produced high-quality results for road splits/merges.

I also contributed to program verification in other parts of the platform. I introduced the synchronous model for control systems, later used by other teams to verify the vehicle interface.

Standard Chartered Bank London, UK

Quantitative Developer Oct 2018 - Mar 2019

I was part of the risk analytics team and wrote checks for the bank's value-at-risk estimate.

Cog Systems Sydney, Australia

Senior Software Engineer (Contract) Jun 2018 - Sep 2018

I built a toolchain for trusted control systems and protocols.

- I designed and built the front-end specification language, a simplified version of the modelling tool Promela/SPIN. I wrote examples of simple protocols and algorithms in this language to check its expressivity for Cog applications.
- I wrote a compiler to produce verified binaries for programs specified in the front-end language. I targeted the subset of C supported by the certifying C compiler CompCert.
- I wrote a tool to extract proof obligations in Isabelle/HOL from the same front-end program. A user could complete the proofs and gain confidence that the compiled binaries behave as the proof specified.

Ambiata Sydney, Australia

Software Engineer Jun 2015 - Mar 2018

I worked on the Extract-Transform-Load pipeline, which processed terabytes of unstructured data for customers in insurance and advertising.

- I built the compiler for icicle, a *streaming query language*. Icicle guaranteed that valid queries complete successfully, and that input streams are processed only once. I wrote the code generation to C, and the type-checker to enforce those guarantees.
- I tested and fixed bugs in a number of related projects, mainly zebra, a columnar binary data store, and ivory, an immutable feature store.

Anchor Sydney, Australia

Software Engineer Aug 2014 - Apr 2015

I worked on cloud and web services on top of OpenStack.

– I wrote a tool to aggregate customer billing information from counters and metrics across the cluster.

– I wrote telemetry tools for an internal distributed data store, producing profiling information to identify performance bottlenecks in the store.

– I wrote a tool to propagate non-conflicting changes across stores of structured data. This eliminated the need for ad-hoc scripts to reconcile large amounts of JSON/XML produced by other tools.

Open Kernel Labs Sydney, Australia

Software Engineer Mar 2013 - Jun 2014

I worked hypervisor-based virtualization for secure embedded devices.

– I implemented support for virtual hardware interrupts using extensions from ARMv7, to achieve faster virtualization over the existing software virtualization method.

– I worked on the internal build tool to propagate changes across different Linux kernel versions.

National ICT Australia Sydney, Australia

Research Assistant 2011 - 2012

I was an undergraduate student in the seL4 verified microkernel project. I worked on information flow proofs and proof automation tactics.

Education

2009 - 2013 BSc. Computer Science at **University of New South Wales, Australia** (Honours)

Skills

Machine Learning	Deep Learning, Computer Vision
Compiler Engineering	Code Generation, Program Optimizations, EDSLs
Libraries and Tools	PyTorch, TensorFlow, scikit-learn
GPU Programming	OpenCL
Languages	Python, C, Scala, Java, Haskell