Research Statement

Hashim Sharif

With the slowdown of Moore's law and the end of Denard scaling, the gap between compute hardware performance and the computational demands of software applications is widening. Applications deployed on the edge such as deep learning inference, real-time data analytics, real-time robotics tasks, and augmented and virtual reality (AR/VR) are pushing the compute capabilities of current systems. To bridge this gap, I work on techniques that improve the performance and energy-efficiency of software programs by using accuracy-aware optimizations that trade off small amounts of computational accuracy in exchange for significant performance improvements. My PhD dissertation focuses on building **retargetable compiler and runtime systems** that automatically and efficiently navigate the tradeoff space of accuracy and performance offered by the various hardware and software approximation choices available on heterogeneous compute platforms. A major focus of my work is on developing **easy-to-use systems for compute-efficient deep learning at the edge**. I closely engage with industry, large companies and startups, and take a keen interest in releasing open-source systems. As future work, I plan to a) investigate a *configurable and customizable compiler ecosystem* that enables domain experts to create compiler frameworks for new application domains, and b) explore systems for efficient ML on *autonomous cyber-physical systems*.

1 RESEARCH OVERVIEW

Computations in many important application domains such as image and video processing, object detection, speech recognition, and data analytics are inherently approximate in the sense that the input data is often derived from noisy sensors or the output results are computed with some acceptably accurate solutions. This error tolerance property of applications has led to the adoption of *approximate computing techniques*: software and hardware optimizations that trade off small amounts of accuracy for better performance.

Research Challenges. The diversity of approximation techniques and the varying error tolerance across applications introduces multiple challenges: a) naively mapping kernels/operations to their approximate variants may lead to unacceptably low output quality, b) it is unclear how much accuracy can be reduced for individual components (e.g., neural network used in an autonomous robot), without compromising on the end-to-end task quality (e.g., robot navigation), c) different compute units (CPUs, GPUs, accelerators) provide different approximation choices and different accuracy-performance tradeoffs, d) the variety of software and hardware approximation choices introduce a large search space, and hence intelligent exploration techniques are needed, and e) it is difficult to maintain object code portability, while supporting hardware-specific approximation choices.

Summary of Research. To address these challenges, my research explores *three major thrusts* towards sophisticated compiler support for *end-to-end accuracy-aware program optimization*, with the goal of enabling compute-intensive workloads to run efficiently on resource-constrained *heterogeneous edge compute systems*.

- I proposed *ApproxHPVM* [7], the first compiler framework that supports a notion of *hardware-agnostic and portable accuracy metrics* at the Intermediate Representation (IR) level, and uses these metrics for accuracy-aware program optimization.
- I designed *ApproxTuner* [8], a compiler and runtime system that introduces efficient approximation-tuning strategies that scale well for large tensor-based programs, while supporting multiple software and hardware approximation choices for each tensor operation.
- I proposed *ApproxCaliper* [9], the first framework that automatically evaluates the potential for *relaxing accuracy requirements* for neural network models used in an end-to-end application context, and uses this error slack to more aggressively optimize computationally-expensive models.

Industry Collaborations. I work closely with industry to gain insights into emerging real-world workloads and to better identify open research problems that are of direct relevance to production systems. In the past few years, I have actively worked on the DARPA-funded DSSoC (Domain-specific System on chip) project [2] that is led by IBM Research and involves Illinois, Harvard, and Columbia. For DARPA reviews, I have led research teams of PhD students working on the HPVM compiler (project that includes ApproxHPVM and ApproxTuner) with the goal of compiling energy-efficient programs for an application-customized heterogeneous SoC. Last year, I closely collaborated with a robotics startup, Earthsense, to optimize the software stack of a production-ready autonomous agriculture robot and performed field experiments. Our optimizations *reduced the cost of deployed compute hardware by 3×*. I am also keenly interested in open-sourcing my research. I led two releases of HPVM; the first release was presented at the open-source conference FOSDEM 2020 [3]. These open-source releases have facilitated researchers at IBM Research, Harvard, the University of Belgrade, and the University of Ljubljana among others, to adopt HPVM in their own projects.

2 APPROXHPVM: A PORTABLE COMPILER IR FOR ACCURACY-AWARE OPTIMIZATIONS

Concept: ApproxHPVM [7] is a compiler IR and framework that supports accuracy-aware optimization for heterogeneous platforms with diverse compute units and approximation choices, while maintaining the portability of the software package across hardware architectures.
Approach: ApproxHPVM is easy-to-use; it takes as input a program and an accuracy requirement (e.g., classification accuracy), and automatically maps the kernels/operations in the program to different approximation choices and different hardware compute units to maximize performance/energy benefits, while meeting the accuracy requirement. To maintain software portability, ApproxHPVM takes a novel approach by decoupling the approximation selection task into i) a hardware-independent autotuning phase at development-time (before code is shipped) that uses randomized error injection to measure error sensitivity of individual operations, and assigns error budgets in terms of accuracy metrics, and ii) a subsequent install-time (on device) that selects approximation knobs based on these error budgets.
Results: Evaluations using nine popular convolutional neural networks (CNNs) on a heterogeneous edge SoC, ApproxHPVM achieves speedups between 1-9× and energy reductions between 1.1-11.3×, with only one percentage point accuracy loss.
Impact: ApproxHPVM was published at OOPSLA 2019 [7]. To compile programs for the EPOCHs SoC designed in our DARPA-funded DSSoC project, we use ApproxHPVM since it supports easy-to-use frontends for high-level domain-specific languages, a retargetable IR, and efficient code generation backends for accelerators and general-purpose compute units.

3 APPROXTUNER: A COMPILER AND RUNTIME SYSTEM FOR ADAPTIVE APPROXIMATIONS

Concept: ApproxTuner [8] (also part of the ApproxHPVM ecosystem) is an extensible and efficient approximation-tuning framework that supports both static and dynamic (runtime) tuning of approximation knobs.
Approach: ApproxTuner adopts a novel 3-phase strategy for selecting and tuning approximations. It splits tuning into: 1) construction of tradeoff curves for hardware-independent approximations at development-time, 2) mapping to hardware-specific approximations at install-time, and 3) a fast approximation selection at runtime. ApproxTuner introduces *predictive tuning*; a novel model-based approximation-tuning technique that uses compositional models for accuracy prediction to speed up both development- and install-time tuning.
Results: Using hardware-independent approximations on GPU, for 10 popular CNNs and an image processing benchmark, ApproxTuner achieves a mean speedup of 2.2× and mean energy reduction of 2.1×, with merely 1 percentage point accuracy loss. Predictive tuning is 12.8× faster compared to conventional empirical tuning (used in ApproxHPVM), while achieving comparable benefits.
Impact: ApproxTuner is published at PPoPP'21 [8]. ApproxTuner is open-source [4] and is easily extensible to new approximation choices and new hardware backends. ApproxTuner inspired mobile computing researchers at the University of Ljubljana, Slovenia, to extend ApproxTuner with dynamic tuning support for Android systems. Amazon AWS is interested in evaluating ApproxTuner for optimizing machine learning models in the SageMaker framework [1].

4 APPROXCALIPER: EXPLOITING APPLICATION-LEVEL ERROR RESILIENCY FOR OPTIMIZING NEURAL NETWORKS

Concept: We develop ApproxCaliper [9], the first framework that automatically evaluates the potential for optimizing multiple neural network components in an *end-to-end application context.*
Approach: ApproxCaliper analyzes how the end-to-end application quality metric is affected by changes in the neural network accuracy and automatically optimizes the neural network components, while satisfying the end-to-end quality constraints. A calibration phase uses *statistical error injection* in the outputs of the neural network (NN) sub-component(s) to quantify the *minimum* neural network accuracy required to satisfy application-level quality targets. A tuning phase uses these error constraints and optimizes the NN components. To generate a tradeoff space of models to choose from, it uses structured weight pruning.
Results: We evaluate a production-ready autonomous agriculture monitoring robot (developed by Earthsense) and a lane-following golf cart simulator; both use convolutional neural networks for visual perception. ApproxCaliper achieves speedups of up to 5.3× for the agriculture robot, and 2.2× for the lane-following

simulator, without affecting the quality of autonomous navigation. ApproxCaliper is extensible to other kinds of neural network optimizations.

Impact: Working with Earthsense, we optimized their software stack using ApproxCaliper. The performance improvements achieved via ApproxCaliper allowed us to replace a $876 Intel NUC compute device on the production robot with a $35 Raspberry Pi4 that provides the same level of navigation quality. Pi4 also compares favorably to the $99 Jetson Nano, the cheapest device we found to deliver necessary performance without approximations. Our work has led Earthsense to consider lower cost alternatives for compute hardware. ApproxCaliper is currently under review at SIGMETRICS'22.

5 TRIMMER: APPLICATION SPECIALIZATION FOR CODE DEBLOATING

While my PhD dissertation work focuses on accuracy-aware program optimization frameworks, my earlier research focused on *code size reduction techniques*. Minimizing code size is an essential goal for edge computing since many edge and IoT compute devices have limited memory and secondary storage. An opportunity for reducing program size is that libraries and programs often support a more extensive set of APIs and features than needed in a specific usage context. My earlier research proposed TRIMMER [5, 6], a code size reduction tool that specializes programs for user-provided constant configuration parameters, and removes support for unused features. TRIMMER uses novel constant propagation techniques to forward constant values throughout the program call-graph, uses partial evaluation techniques to specialize functions for constant arguments, and removes code that is provably dead for the constant values. For 20 commonly-used Linux programs and utilities, TRIMMER achieves a mean binary size reduction of 23% and a maximum reduction of 63%.

6 FUTURE WORK

My research on compiler and runtime systems for end-to-end accuracy-aware program optimization shows that good system support enables significant performance and energy improvement for real-world workloads. In the future, my goal is to develop new systems that make it easy to exploit accuracy-aware optimizations across the computing stack; from algorithms to approximate hardware. I am excited about exploring a configurable and customizable compiler ecosystem that enables domain experts to create compiler frameworks for new application domains (Section 6.1). I am also interested in exploring systems for efficient ML on autonomous cyber-physical systems (Section 6.2).

6.1 Customizable Compiler Systems for Energy-efficient Computing at the Edge

An increasing number of application domains are being deployed at the edge. Some of these include augmented and virtual reality (AR/VR), machine learning, data analytics, image and video processing. With this increasing diversity in applications, compiler systems need to support a broad range of algorithms and operator types.

To enable energy-efficient computing for a wide range of domains, I envision a flexible compiler development ecosystem that domain experts can use to configure custom compilers for new application domains. Making it easier for domain experts to develop and extend compilers will help facilitate energy-efficient computing at the edge. Towards this goal, I envision these opportunities:

- **Domain-specific Optimization Opportunities.** I envision compiler frameworks that give users the flexibility to provide a list of domain-specific transformations in high-level specifications, while the compiler automatically discovers configurations of transformations that maximize energy-efficiency and/or performance. For instance, experts can specify domain-specific rules for operator fusion, and the compiler can automatically identify program-specific fusion opportunities that improve performance.
- **Discovering Algorithms with Accuracy-Performance Tradeoffs.** A non-trivial amount of developer effort is required to identify low-level software and algorithmic knobs that offer accuracy-performance tradeoffs. As compiler support is extended to new domains, we need mechanisms that exploit operator- and algorithm-specific approximation opportunities with minimal supervision. I will investigate compiler techniques that *automatically discover approximate variants for algorithms* that developers specify in high-level domain-specific languages (DSLs).
- **Efficient Code Generation Frameworks.** Edge computing platforms are becoming increasingly heterogeneous. Many edge devices incorporate domain-specific hardware accelerators that support knobs for controlling accuracy, performance, and energy tradeoffs. I will explore efficient code generation frameworks

that exploit the full potential for accuracy-aware optimization by simultaneously tuning software and hardware approximation knobs. Moreover, since the choice of hardware resource usage parameters (e.g., tile sizes, scratchpad usage) significantly impacts performance, I envision these systems co-tuning hardware resource parameters with algorithmic knobs for approximation.
- **Enabling Energy-efficient Robust Systems.** For safety critical domains such as autonomous driving, and medical diagnosis, systems must be robust and secure in adversarial environments. I will investigate compiler techniques that automatically navigate the inherent tradeoffs of energy, accuracy, and robustness of edge systems.

For this line of research, I plan to collaborate with experts in emerging edge-focused application domains such as augmented and virtual reality (AR/VR), natural language processing, and deep learning.

6.2 Systems for Efficient ML on Autonomous Cyber-physical Systems

Autonomous robots, drones, and vehicles (AVs) are increasingly employing deep neural network (DNN) pipelines for on-device inference tasks. Running these compute-heavy inference tasks is a significant barrier in adopting learning-based methods in cost-constrained domains, such as agriculture, forestry, delivery, and mining. My goal is to design compiler and runtime systems that make it possible to deploy sophisticated machine learning and control algorithms on such cost- and energy-constrained cyber-physical systems. Towards this goal, I see these opportunities:

- **Easy-to-Program Systems**. I will investigate *end-to-end compiler systems* that will make it easier for machine learning and robotics experts to program cyber-physical systems. I envision this to incorporate: a) novel *easy-to-program* domain-specific languages that facilitate designing perception and control algorithms, b) *easy-to-configure* optimization frameworks that allow experts to plug in domain- and application-specific knowledge, and c) *accuracy-aware code optimization* frameworks that automatically navigate quality, performance, and energy tradeoffs for algorithms and policy choices specified by domain experts.
- **Task-aware Optimization Strategies.** Autonomous planning and control systems usually include multiple components that process and fuse different sensor modalities, including cameras, LIDAR, IMU, and range sensors. I will investigate optimization systems that use *task-aware optimization strategies* to co-tune machine learning models and other application components. I believe such systems will drive far more significant benefits compared to tuning components in isolation.
- **Application-Specific Dynamically Adaptable Systems.** ApproxTuner [8] supports accuracy-aware runtime tuning that counteracts slowdowns imposed by low-power and low-frequency modes. In the future, I will explore customizable runtime systems that allow users to customize application-specific dynamic control policies, while abstracting away model optimizations and system tuning that enables these adaptations. For instance, users should be able to specify situations that require high-quality configurations and others than require low energy consumption at the cost of some user-given acceptable loss in quality.
- **Optimizing Cooperative Multi-Agent Robot Systems.** Cooperative multi-agent robot systems are being used in many domains, such as search and rescue, surveillance systems, and digital agriculture. I want to explore compiler and runtime techniques that use computation and communication approximation techniques to optimize the overall energy consumption and compute performance for a robot swarm, while maintaining acceptable task quality.

In my PhD, I closely collaborated with Earthsense, a robotics startup that develops small autonomous mobile robots for agriculture monitoring. As a faculty member, I plan to collaborate with researchers working on various kinds of autonomous cyber-physical systems.

REFERENCES

[1] 2021. Amazon SageMaker. https://aws.amazon.com/sagemaker/

[2] 2021. Domain-Specific System on Chip (DSSoC) Program. https://eri-summit.darpa.mil/docs/20180725_1100_DSSoC.pdf

[3] 2021. HPVM: Extending LLVM For Compiling to Heterogeneous Parallel Systems. https://archive.fosdem.org/2020/schedule/event/llvm_hpvm/

[4] 2021. HPVM Public Release. https://gitlab.engr.illinois.edu/llvm/hpvm-release

[5] Aatira Anum Ahmad, Abdul Rafae Noor, **Hashim Sharif**, Usama Hameed, Shoaib Asif, Mubashir Anwar, Ashish Gehani, Junaid Haroon Siddiqui, and Fareed M Zaffar. 2021. TRIMMER: An Automated System for Configuration-based Software Debloating. *IEEE Transactions on Software Engineering **(TSE'21)***.

[6] **Hashim Sharif**, Muhammad Abubakar, Ashish Gehani, and Fareed Zaffar. 2018. TRIMMER: Application Specialization for Code Debloating . ***(ASE'18)*** .

[7] **Hashim Sharif**, Prakalp Srivastava, Muhammad Huzaifa, Maria Kotsifakou, Keyur Joshi, Yasmin Sarita, Nathan Zhao, Vikram S. Adve, Sasa Misailovic, and Sarita Adve. 2019. ApproxHPVM: A Portable Compiler IR for Accuracy-aware Optimizations. ***(OOPSLA'19)*** .

[8] **Hashim Sharif**, Yifan Zhao, Maria Kotsifakou, Akash Kothari, Ben Schreiber, Elizabeth Wang, Yasmin Sarita, Nathan Zhao, Keyur Joshi, Vikram Adve, Sasa Misailovic, and Sarita Adve. 2021. ApproxTuner: A Compiler and Runtime System for Adaptive Approximations. ***(PPoPP'21)*** .

[9] **Hashim Sharif**, Yifan Zhao, Peter Pao-Huang, Vatsin Ninad Shah, Arun Narenthiran, Mateus Valverde Gasparino, Nathan Zhao, Abdulrahman Mahmoud, Sarita Adve, Girish Chowdhary, Sasa Misailovic, and Vikram Adve. 2022. ApproxCaliper: Exploiting Application-level Error Resiliency for Optimizing Neural Networks. Under Review at *ACM SIGMETRICS 2022*.