Autonomous Realization of Safety- and Time-Critical Embedded Artificial Intelligence

Joakim Lindén*§, Andreas Ermedahl†‖, Hans Salomonsson‡,
Masoud Daneshtalab§, Björn Forsberg¶, Paris Carbone‖¶
*Department of Aeronautical Engineering, SAAB AB, Linköping, Sweden, joakim.linden@saabgroup.com
†Network Division, Ericsson AB, Kista, Sweden, andreas.ermedahl@ericsson.com
‡EMBEDL AB, Gothenburg, Sweden, hans.salomonsson@embedl.com
§School of Innovation, Design and Engineering, Mälardalens Universitet, Västerås, Sweden, masoud.daneshtalab@mdu.se
¶Department of Computer Science, RISE Research Institutes of Sweden AB, Kista, Sweden, bjorn.forsberg@ri.se
‖School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Kista, Sweden, parisc@kth.se

***Abstract*—There is an evident need to complement embedded critical control logic with AI inference, but today's AI-capable hardware, software, and processes are primarily targeted towards the needs of cloud-centric actors. Telecom and defense airspace industries, which make heavy use of specialized hardware, face the challenge of manually hand-tuning AI workloads and hardware, presenting an unprecedented cost and complexity due to the diversity and sheer number of deployed instances. Furthermore, embedded AI functionality must not adversely affect real-time and safety requirements of the critical business logic. To address this, end-to-end AI pipelines for critical platforms are needed to automate the adaption of networks to fit into resource-constrained devices under critical and real-time constraints, while remaining interoperable with de-facto standard AI tools and frameworks used in the cloud. We present two industrial applications where such solutions are needed to bring AI to critical and resource-constrained hardware, and a generalized end-to-end AI pipeline that addresses these needs. Crucial steps to realize it are taken in the industry-academia collaborative FASTER-AI project.**

***Index Terms*—machine learning, embedded systems**

I. Introduction

We are undergoing a radical shift in computational systems driven by the fast advances in Artificial Intelligence (AI) and Machine Learning (ML). Today, the de-facto standard processes for classification, regression and complex optimisation are heavily grounded on ML-based inference, driven by cloud-centric actors such as Google and Amazon. The availability of low-cost computing resources for training and inference has made this the standard development and deployment platform for such workloads. To support this growing demand, the frameworks, ML compilers, and serving solutions of end-to-end ML pipelines have become highly geared towards cloud and cloud-vendor specific ML accelerators.

Simultaneously, there is a large amount of installed infrastructure all over the world which could benefit from AI. These systems often rely on resource-constrained and/or otherwise non-commodity hardware specialized for their specific task, and are deployed once with an expected lifetime of many years or decades. Examples include hundreds of thousands of telecom base stations worldwide, based upon Digital Signal Processors (DSP) specially built for high-bandwidth communication workloads, and applications using Field-Programmable Gate Array (FPGA)-based hardware to powering decision making processes and communication functions in aircrafts and other vehicles. This is in contrast to the fast-paced replacement of cloud servers, smart phones, and other devices where ML is ubiquitous, and bringing ML to such systems presents several challenges. Rapid replacement of such infrastructure would lead to both wasteful resource usage in conflict with goals of green and sustainable business practices, and lead to unrealistically high and unnecessary costs. This is especially clear in the need to re-architect support for all existing critical usage (e.g., signal processing, event-based logic, etc.) while ensuring critical requirements and constraints remain unviolated, incurring high verification costs. Instead, solutions that enable AI workloads on such systems must be developed, improving their respective applications while simultaneously re-extending their lifetime to what they were originally built for. In doing this, interoperability with cloud-centric existing infrastructure must be preserved to the largest extent possible to avoid fragmentation of the ecosystem.

We foresee a future where ML-serving and critical business logic can co-exist from programming model, via optimizing compilers, to deployment on any specialized hardware. FASTER AI has been conceived to help bring fully automated support for ML-inference to critical hardware. This will contribute to a smoother transition towards more ML-dominant workloads within widely available critical hardware infrastructure. To effectively bring ML models to resource-constrained hardware we intend to combine cutting-edge neural architecture search methods incorporating hardware constraints, as well as production-grade experience in model compression, coupled with optimized compilation and optimisation expertise. Our effort will enable Swedish ICT leaders SAAB and Ericsson, and others, to take their existing hardware into the AI future. At the same time, FASTER AI's constrained search and intermediate multi-level compilation method is adaptable to any target hardware architecture with minimal effort.

This is an early project concept presentation of the

979-8-3503-4859-0/DATE24/© 2024 EDAA

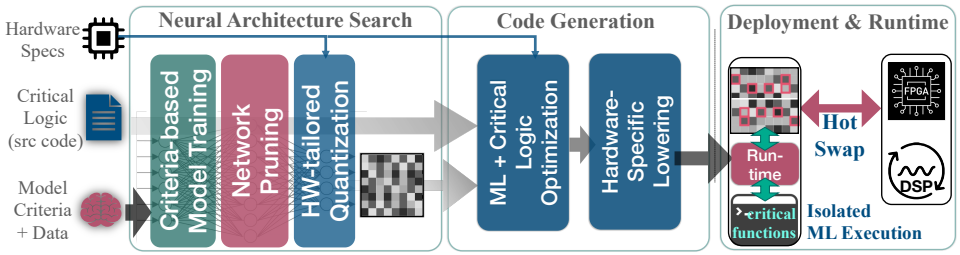

Fig. 1. Proposed AI pipeline to equip specialized hardware with AI capabilities.

FASTER-AI project, funded by Sweden's Innovation Agency. The rest of the paper is organized as follows: Section II presents the identified challenges in avionics and telecom use-cases, Section III presents the proposed solution and methodology of the project, with a summary in Section IV.

II. Background

Critical and specialized hardware systems have become deeply enveloped in modern society, providing fundamental services that we take for granted. Even with their importance, these systems have often been overlooked as the promises of Artificial Intelligence have rapidly started to be fulfilled on other systems and in other parts of society. Key challenges in the development for, and deployment of, ML workloads on critical infrastructure can be summarized as follows:

1) The resource constrained nature of these systems mean that larger models cannot be directly deployed on these devices. Either because of too high memory or storage requirements, or simply because their performance is not adequate on these devices.
2) Many of these systems are built for a special purpose with custom execution environments, compilers, and even programming models catering to their domain. Adding AI workloads to such systems is non-trivial in several levels of the software development and execution stack, including optimizing AI-compilers or even fundamental backend support for popular AI frameworks, and the runtime frameworks that support them.
3) The co-existence of traditional safety-critical logic and AI on complex architectures within highly integrated systems leads to concerns about fail-safe operation and violation of real-time timing guarantees of their original applications. Such timing and safety requirements set them clearly apart from best-effort non-critical and containerized systems typically executed in the cloud.

Solutions to these problems have started to appear – and in some areas are maturing fast – but deployment of ML-workloads on critical infrastructure is still hard and manual labor. Addressing (1), model compression techniques, such as pruning [1] and quantization [2], have been demonstrated to be effective in reducing the memory and computation demands of neural networks, while still retaining a high enough accuracy and precision to make them competitive. However, incorrectly applying these techniques can significantly degrade the performance, and they are more difficult to train. For (2), the lack of tooling is significantly alleviated by the ubiquitous use of open source in the machine learning ecosystem. While many cloud vendors provide their own proprietary frontends, the underlying graph and compiler optimization frameworks are often open source (Pytorch, Tensorflow, MLIR [15], OpenVINO, and many others). FASTER-AI will build upon these to bring cloud-level maturity and compatibility into AI frameworks for critical infrastructure. Still, significant challenges remain, most obviously the need to adapt optimization and code generation frameworks to the non-commodity hardware architectures employed in these systems. Just as important is the ability to co-develop and co-deploy traditional business critical logic with AI, mapping to the development processes of these highly-specialized critical systems. This becomes even more challenging for fail-safety and timing constraints [3] to address (3), also needing execution environment support.

A. Identified Use-Cases

The project takes its main motivation from the industry's desire for AI-deployment in existing critical hardware alongside traditional business logic, providing best-of-both-worlds processing without the need for early hardware replacement. This section presents two applications, one from avionics, and one from telecommunications. These are two industries where traditional business logic can benefit from AI/ML, given that critical real-time properties are preserved and deployment tools are adopted to resource-constrained systems.

Avionics platforms often require latency-critical real-time detection of objects for increased situational awareness and pilot workload reduction. ML model execution efficiency is integral in such processing chains and specialized hardware solutions are required for current and future fast-paced sensor data acquisition demands. The ability to adapt and optimize ML models and adjacent control logic to the available hardware resources and architecture is key for robust and performant operation. At the same time, special care needs to be taken to isolate ML execution and respect functional separation while meeting latency, accuracy and energy requirements. This is paramount in safety-critical avionics applications where guarantees of uninterrupted execution are required.

Telecom base stations have reliably served mobile user-equipment and corresponding requests over the past decades.

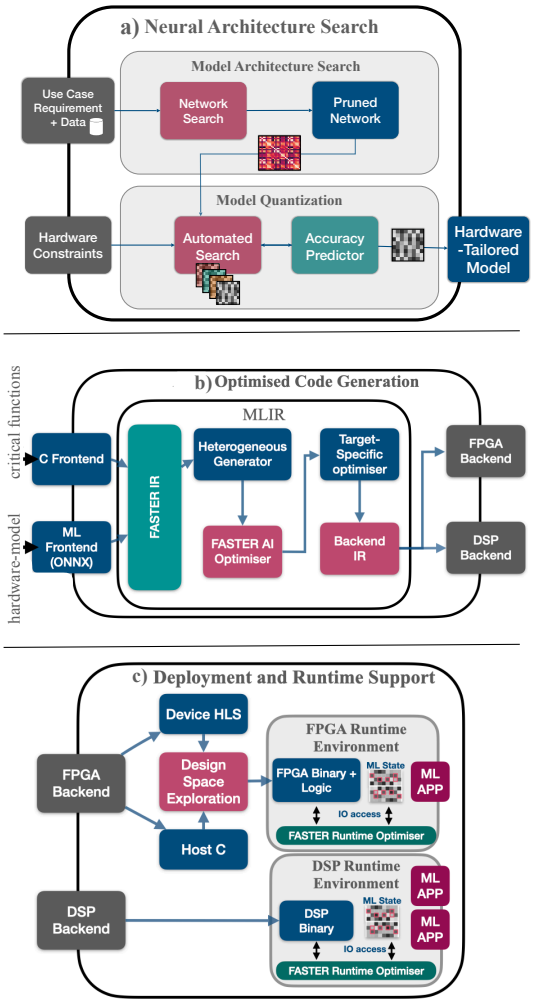

Fig. 2. Detailed methodology Overview

The hardware has been designed with low latency criteria to reliably serve many ten thousands of requests per second at each base station without disruptions. ML is a promising option to optimize select parts of the complex radio resource management problem. Furthermore, any ML model used must be realized with latency-, compute- and memory-constraints in mind given, e.g., 6G requirements. Despite their capabilities, today's base station hardware often lack standardized support for compiler toolchains for ML programming and execution due to their non-conformity to commodity architectures.

III. Proposed Solution

To address these challenges, we propose a end-to-end systemic approach, as shown in Figure 1. The proposed solution is a generalized approach to automate and optimize ML functions for today's critical hardware needs, but is general also for upcoming hardware releases and new application areas. The proposed approach has three phases, Neural Architecture Search, Optimized Code Generation, and Deployment and Runtime Support. This section presents these in detail.

A. Neural Architecture Search

Designing and deploying performant and compact models for resource-constrained systems is a key challenge. Currently most ML models do not fit within resource-constrained devices and any workarounds such as precision cutting fall into the inexperienced hands of system developers. Network architecture design and model compression, such as pruning and quantization, have been the two dominant research directions, but both impose excessive time costs and extensive hyperparameter tuning. Several recent works [5]–[11] considered automated methods to search for the best network architecture tailored to constrained hardware, or to prune and quantize the DNNs. PR-DARTS [5] showed a differentiable neural architecture search method that jointly optimizes the network architecture and pruning strategy. TAS [6] integrates quantization into neural architecture search by introducing a new cell template for one-shot NAS with a learnable quantizer adaptively relaxing the quantization mechanism. These automation processes address network architecture, pruning, and quantization in isolation, and independently applying these gives sub-optimal results [8].

A systemic approach must include hardware-aware network architecture and compression search in unison to co-optimize all these aspects for higher performance neural networks. FASTER AI as shown in Figure 2(a) combines both of these techniques as a first step in each ML deployment. To this end, the main important stage is evaluating the accuracy of different network architectures with different bit widths in the quantization method. By evaluating the accuracy of the quantized network, we can use an automated (e.g., evolutionary) search algorithm to find the best pruned and quantized network architecture. Specifically, in the first step, we train a super network that supports a diversity of pruning ratios by adding pruned operations to the search space. Then, we quantize each sub-network from this large network and evaluate its accuracy. Finally, we will utilize an automated search algorithm with different hardware constraints to find the best quantized network [9], imposed by the use-case requirements. To avoid high search times in complex sub-network quantization, we train a simple feed-forward neural network based on collecting a data set from important features of quantized networks (such as kernel size, channel numbers, weight/activation bits, etc.) to predict the accuracy of the quantized network. This prediction can be used to accelerate the search process.

B. Optimized Code Generation

Many tools for compilation and optimization of ML workloads for different hardware have been developed in recent years, e.g., MLIR [15], XLA [16], and TVM [14], [17]. While some of them target common resource- constrained systems, they fail to address the specific challenges in integrating them with co-executing critical logic on specialized hardware. To avoid duplicating efforts, the systemic approach must build upon reusable and extensible compiler solutions established in the general-purpose or cloud-oriented world, and be adapted to address the specific challenges of real-time critical systems. Solutions based on e.g., MLIR [15], extended to address the

specific real-time challenges are a promising direction. MLIR's composability enables reuse of existing ML optimizations, allowing efforts to be concentrated on challenges specific to critical and resource-constrained hardware.

The compiler of FASTER AI as shown in Figure 2(b) will build upon open source frameworks, including MLIR, and be the critical bridge linking the ML networks and the critical logic to the runtime and middleware available on the target hardware platform. To facilitate this, we will build a FASTER AI dialect using MLIR to generate optimized and integrated ML and critical logic code, targeting the runtime environment of the project platforms. We further intend to generate synthesizable code for FPGA platforms, by lowering the FPGA platform code to C/C++ which can be passed to the hardware vendor-specific High-Level Synthesis (HLS) tools. In the longer term we envision lowering code which targets FPGAs directly to hardware description languages inspired by currently experimental MLIR dialects such as CIRCT [20], or use alternative approaches such as Xilinx' AIEngine [21]. The output of the FASTER code generation is the optimized program, either in executable format or in C/C++ code, targeting the architecture and runtime system of each platform. Focusing efforts towards a widely adopted optimization and compilation framework will create a future-proof code generator for end-to-end AI pipelines for critical resource constrained systems.

C. Deployment and Runtime Support

A final challenge is AI execution and co-existence with critical functions on the hardware. Deployment of models to FPGA platforms is particularly troublesome. While the ability of AI/ML frameworks to generate Register Transfer Level (RTL) designs have evolved in recent years [12], they have not yet been exploited properly, as exploring and identifying appropriate synthesis directives needs both time and expertise, imposing significant engineering effort. Automatic Design Space Exploration tools to analyze RTL design latency and resource utilization is required to optimize the characteristics of each FPGA kernel. Furthermore, in resource-constrained systems the use of model hot-swapping [13] significantly improves power and resource utilization, by properly time-multiplexing functionality in adaptive and scalable kernels with respect to fabric and time-slice availability and mission constraints. A systemic approach should provide a portable runtime to manage ML functions to not violate the functional separation, nor latency, accuracy, or energy requirements of critical code while executing on the platform.

To achieve this in FASTER-AI we plan to orchestrate the kernel reconfiguration at runtime, taking into account that the reconfiguration latency should be hidden by the scheduler to preserve the actual desired throughput. Figure 2(c) shows a more detailed overview of this endeavor. To further address the safety-critical aspect of the use-cases, special care will be taken to limit points of potential interference and promote isolation between co-executing processes.

IV. The FASTER-AI project

FASTER-AI addresses these challenges by accelerating data-driven innovation and AI by enabling ML on non-commodity yet capable heterogeneous hardware used widely in leading Swedish industries – aerospace and telecommunications. We will measure the increase in computational and energy efficiency and ML performance on targeted hardware, with benefits also effective in other domains of societal and environmental importance. FASTER-AI builds on open-source technologies and state of the art research results to construct model search and code generation pipelines for non-commodity hardware architectures, granting longer life-span to existing hardware via native ML-inference. It furthers the possibilities of transferring our optimized approach to upcoming hardware by automating and optimizing the integration of ML functions to meet the needs of critical applications on critical hardware. Close collaboration between leading Swedish industry and institutes enables young researchers to contribute to knowledge transfer across academia and industry.

FASTER-AI is supported by the Swedish Innovation Agency (2022-03036) and supercomputing resource Berzelius provided by the National Supercomputer Centre at Linköping University and the Knut and Alice Wallenberg Foundation.

References

[1] V. Sehwag, et al., "Hydra: Pruning adversarially robust neural networks," Advances in Neural Information Processing Systems, 2020.
[2] Babak Rokh, et al., "A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification, ACM, 2023.
[3] A. Luppold, et al., "Compiling for the Worst Case: Memory Allocation for Multi-task and Multi-core Hard Real-time Systems," ACM, 2020.
[4] H. Mousavi, et al., "DASS: Differentiable Architecture Search for Sparse Neural Networks", ACM, 2023.
[5] H. Mousavi, et al, "PR-DARTS: Pruning-Based Differentiable Architecture Search," arXiv preprint arXiv:2207.06968 (2022).
[6] M. Loni, et al., "TAS: Ternarized Neural Architecture Search for Resource-Constrained Edge Devices." (DATE). IEEE, 2022.
[7] A. Burrello, et al., "Enhancing Neural Architecture Search with Multiple Hardware Constraints for Deep Learning Model Deployment on Tiny IoT Device", IEEE, 2023
[8] J. Lin, et al., "Mcunet: Tiny deep learning on iot devices." Advances in Neural Information Processing Systems 33 (2020): 11711-11722.
[9] T. Wang, et al., "Apq: Joint search for network architecture, pruning and quantization policy." IEEE/CVF 2020.
[10] I. Fedorov, et al., "UDC: Unified DNAS for Compressible TinyML Models." arXiv preprint arXiv:2201.05842 (2022).
[11] M. Risso et al., "Lightweight Neural Architecture Search for Temporal Convolutional Networks at the Edge,", IEEE, 2023.
[12] H. Ye et al., "Scalehls: A new scalable high-level synthesis framework on multi-level intermediate representation," IEEE HPCA 2022.
[13] C. Dahlstrom, "Hot swapping Core ML models on the iPhone," Zedge, 2017.
[14] T. Chen, et al., "TVM: end-to-end optimization stack for deep learning." 13th USENIX OSDI, 2018.
[15] C. Lattner et al., "MLIR: A compiler infrastructure for the end of Moore's law," 2020, arXiv:2002.11054.
[16] C. Leary and T. Wang, "XLA: Tensorflow, compiled," TensorFlow Dev Summit, 2017.
[17] microTVM: TVM on bare-metal, https://tvm.apache.org/docs/topic/microtvm/index.html
[18] J. E. Stone et al., "Opencl: A parallel programming standard for heterogeneous computing systems," , 2010.
[19] Mohammad Riazati et al.. "DeepHLS: A complete toolchain for automatic synthesis of deep neural networks to FPGA.", IEEE, 2020.
[20] CIRCT Circuit IR Compilers and Tools, https://circt.llvm.org/.
[21] Xilinx AIEngine on Github, https://xilinx.github.io/mlir-aie/.