DEPARTMENT OF COMPUTER SCIENCE | LUNDS TEKNISKA HÖGSKOLA | PRESENTED 2024-01-09

MASTER'S THESIS MLIR-based Code Generation for High-Performance Machine Learning on AArch64
STUDENT Johanna Gustafson
SUPERVISORS Jonas Skeppstedt (LTH), Fredrik Knutsson (Arm), Per Åstrand (Arm)
EXAMINER Flavius Gruian (LTH)

Using MLIR to conquer machine learning

POPULAR SCIENCE SUMMARY **Johanna Gustafson**

In the quest of high execution speeds, current machine learning systems tend to rely on high-performance compute libraries optimized for a narrow range of hardware devices. Thus, we propose a modular approach in the design of machine learning systems, utilizing the reusable and extensible MLIR compiler framework.

Imagine you are in middle school, given the assignment of constructing a durable miniature bridge. The more weight it can bear, the better. Your teacher explains that the best bridges tend to be based upon triangular units and mentions the *Howe truss* and *Warren truss* bridges. With no further explanation, they walk out of the room. Suddenly, you feel confused; what is Howe or Warren truss? Not only do you have to come up with a design through guesswork, you also have to collect material on your own.

Our master's thesis investigates this issue, although replacing bridges with machine learning systems and durability with performance. More specifically, the problem pertains to adapting advancements within machine learning technology to the ever-expanding plethora of hardware. We believe MLIR (Multi-Level Intermediate Representation) — a novel approach to building reusable and extensible compiler infrastructure — is a solution to this issue. MLIR can be likened to the teacher providing you and your classmates with popsicle sticks, glue and a suggested step-by-step guide on how to go about the task, although giving you the freedom to modify and expand as you wish. This simple consideration eliminates the effort by you and your classmates to each surpass the initial threshold of the assignment.

Given the width of machine learning topics, we focused on one fundamental computation — multiplication and addition of matrices — and demonstrated the implementation of an optimized MLIR operation, as presented in figure 1. We believe the results of our research is indicative of the potential benefits of incorporating MLIR into machine learning systems, in constituting retargetability while retaining high performance.

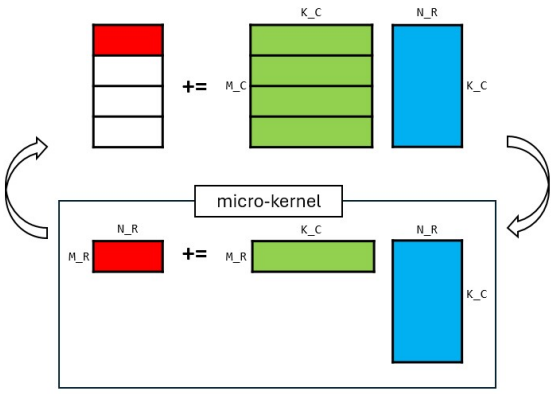

Figure 1: The implemented operation is essentially a nest of loops around a so-called *micro-kernel*; a small sequence of computation tasks. This image illustrates the two innermost loops, where partitions of the three original matrices are multiplied and added.