muRISC-V-NN: Improving RISC-V Vector Extension Performance with a Kernel Library

Jefferson Parker Jones, Philipp van Kempen, Daniel Mueller-Gritschneder, Ulf Schlichtmann

Motivation

• The RISC-V ISA provides a set of vector extensions [1] featuring powerful SIMD instructions which can be used to accelerate machine learning tasks on edge devices.
• The RISC-V ecosystem lacks a lightweight, open-source, and vendor-agnostic compute library to support these extensions.

State of the Art

• The CMSIS-NN project provides kernels optimized for ARM processors with Neon or Helium, Arm’s vector processing extensions [3].
• Autovectorization which is normally applied by embedded software compilers such as the LLVM project and RISC-V GNU Toolchain is a challenging task [8].
• A number of open-source RVV implementations have been published, however none currently exists that implements the full RVV 1.0 specification [4, 5, 6].

Goals

• Propose an optimized kernel library, muRISC-V-NN, which fills the gap in current industry standard ML deployment toolchains for the RISC-V ecosystem.
• Integrate muRISC-V-NN into Tensorflow Lite (TFLite) for Microcontrollers (TFLM) [2] as a bit accurate replacement for CMSIS-NN.
• Demonstrate the effectiveness of muRISC-V-NN by comparison with the default reference kernels and optimized scalar kernels that have been auto-vectorized by the LLVM compiler.

Optimized Reference Kernels

CMSIS-NN [3]

• Use CMSIS-NN kernels as baseline for muRISC-V-NN
• Provides basic implementation
• Interface with TFlite by mapping function names
• Helps to maintain bit-accuracy

RISC Vector Extension Features:

• 32 Vector Registers VL bits long
• Element size (SEW) is dynamic
• Vectors can span multiple vector registers through grouping (LMUL)
• Max number of elements in a vector (LMUL • VREGs)

Designing and Optimizing Kernels:

• Kernels are written to be VL agnostic
• Select SEW and LMUL based on expected parallelism
• Adjust degree of loop unrolling to maximize use of the vector registers
• Prevent Register Spill
• Design tradeoff between LMUL and Loop unrolling

Vectorization Challenges:

• Precision of kernel inputs forces a minimum SEW
• TFLM memory layout complicates kernels
• Im2Col [3] transform helps, but increases memory usage
• Depthwise Convolutions difficult to vectorize
• Filters often small (3x3 or 5x5)
• Number of channels often much smaller than maximum vector length

References


Experimental Results

MLPerf Tiny Benchmark [7]

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use Case</th>
<th>Quantized Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>toycar</td>
<td>LNN</td>
<td>Anomaly Detection</td>
<td>270 kB</td>
</tr>
<tr>
<td>resnet</td>
<td>CNN</td>
<td>Image Classification</td>
<td>96.2 kB</td>
</tr>
<tr>
<td>vww</td>
<td>CNN</td>
<td>Visual Wake Words</td>
<td>325 kB</td>
</tr>
<tr>
<td>awv</td>
<td>CNN</td>
<td>Keyword Spotting</td>
<td>58.3 kB</td>
</tr>
</tbody>
</table>

Experimental Setup:

• Target Architecture: riscv32imve32x
• Deployment Framework: TFLM
• Models Quantized to 8-bit integer
• riscv-isa-sim instruction set simulator used for simulation
• muRISC-V-NN compared directly to TFLM default and muRISC-V-NN scalar kernels
• muRISC-V-NN compared to same scalar kernels auto-vectorized by LLVM

Results:

Performance

VREG0 0 1 2 3
VL = 128 bits, SEW = 16 bits, LMUL = 1

Vectorization Challenges:

• Design tradeoff between LMUL and Loop Unrolling
• Interface with TFlite by mapping function names
• Provides basic implementation
• Helps to maintain bit-accuracy

Optimized Reference Kernels

CMSIS-NN [3]

• Use CMSIS-NN kernels as baseline for muRISC-V-NN
• Provides basic implementation

RISC Vector Extension Features:

• 32 Vector Registers VL bits long
• Element size (SEW) is dynamic

Designing and Optimizing Kernels:

• Kernels are written to be VL agnostic
• Select SEW and LMUL based on expected parallelism

Observations:

• Performance increase of 2x or greater over the default TFLM kernels
• On average, 27.5% to 38.2% faster than autovectorized kernels for the smallest and largest vector lengths respectively

Future Work

• Support for sub-byte and mixed precision models
• Compile-time selection between kernels optimized for different targets

Contact:

jefferson.jones@tuwien.ac.at
philipp.van-kempen@tum.de
ufschlichtmann@tum.de

Open Source:

https://github.com/tum-vv-research