Chair of Electronic Design Automation
Department of Computer Engineering
School of Computation, Information and Technology
Technical University of Munich

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

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

Problem Statement

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 implments the full RVV 1.0 specifcation [4, 5, 6]

Goals

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

Optimized Reference Kernels

CMSIS-NN [3]
Use CMSIS-NN kernels as baseline for muRISCV-NN

- Provides basic implementation
- Interface with TFlite by mapping function names
- Helps to maintain bit-accuracy

RISCV 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 $(\mathrm{LMUL} \cdot {}^{\mathrm{VL}}/_{\mathrm{SEW}})$

VL = 128 bits, SEW = 32 bits, LMUL = 2

VREG0	**3**	**2**	**1**	**0**
VREG1	**7**	**6**	**5**	**4**

VL = 128 bits, SEW = 16 bits, LMUL = 1

VREG0	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**

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

[1] RISC-V Collaboration. (2021) RISC-V "V" Vector Extension v1.0. url: https://github.com/riscv/riscv-v-spec/tree/v1.0.
[2] Robert David et al. (2021) Tensorflow lite micro: Embedded machine learning for tinyml systems. In: Proceedings of Machine Learning and Systems 3 pp. 800–811.
[3] Liangzhen Lai, Naveen Suda, and Vikas Chandra. (2018) Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. In: arXiv preprint arXiv:1801.06601
[4] Matheus Cavalcante et al. (2019) Ara: A 1-GHz+ scalable and energy-efficient RISC-V vector processor with multiprecision floating-point support in 22-nm FD-SOI. In: IEEE Transactions on Very Large Scale Integration (VLSI) Systems 28.2 pp. 530–543.
[5] Matheus Cavalcante et al. (2022) Spatz: A Compact Vector Processing Unit for High-Performance and Energy-Efficient Shared-L1 Clusters. In: Proceedings of the 41s IEEE/ACM International Conference on Computer-Aided Design. pp. 1–9.
[6] Michael Platzer and Peter Puschner. (2021) "Vicuna: a timing-predictable RISC-V vector coprocessor for scalable parallel computation". In: 33rd euromicro conference on real-time systems (ECRTS 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
[7] Colby Banbury et al. (2021) Mlperf tiny benchmark. In: arXiv preprint arXiv:2106.07597.
[8] Neil Adit and Adrian Sampson. (2022) Performance Left on the Table: An Evalua- tion of Compiler Autovectorization for RISC-V. IEEE Micro 42, 5 (2022), 41–48

Experimental Results

MLPerf Tiny Benchmark [7]

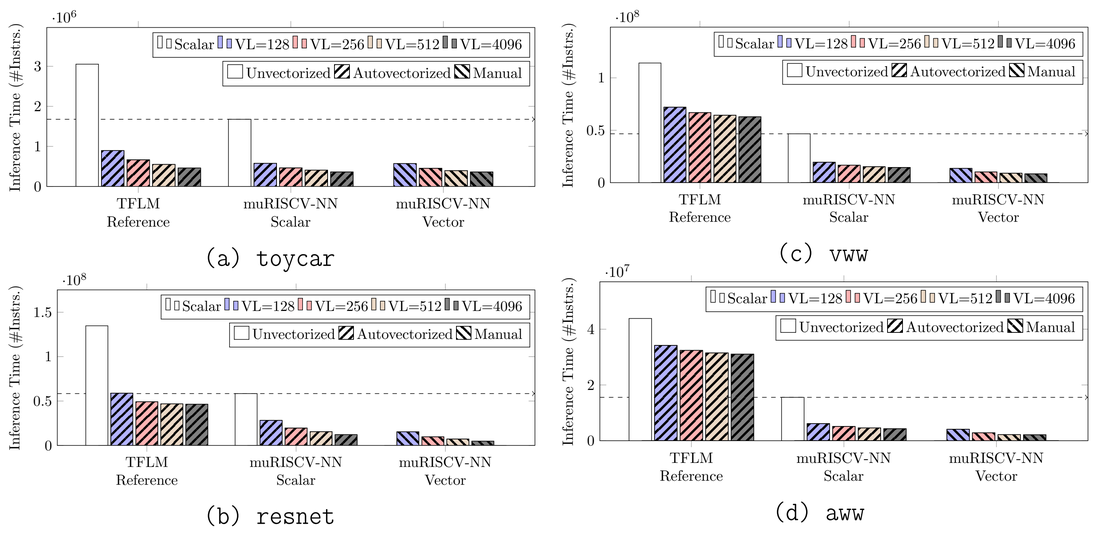
Name	Type	Use Case	Quantized Size
`toycar`	DNN	Anomaly Detection	270 kB
`resnet`	CNN	Image Classification	96.2 kB
`vww`	CNN	Visual Wake Words	325 kB
`aww`	CNN	Keyword Spotting	58.3 kB

Experimental Setup:

- Target Architecture : rv32imzve32x
- Deployment Framework : TFLM
- Models Quantized to 8-bit integer
- `riscv-isa-sim` instruction set simulator used for simulation
- muRISCV-NN compared directly to TFLM default and muRISCV-NN scalar kernels
- muRISCV-NN compared to same scalar kernels autovectorized by LLVM

Results:

Performance



RAM Overheads

	TFLM Reference	muRISCV-NN Scalar	muRISCV-NN Vector
Model	*Arena [kB]*	*△ (%)*	
`toycar`	2.4	+5.0%	+283.3%
`resnet`	54.4	+1.1%	+1.2%
`vww`	100.7	+0.0%	+0.1%
`aww`	22.8	+0.0%	+19.7%

ROM Overheads

	TFLM Reference	muRISCV-NN Scalar		muRISCV-NN Vector
Vect.	-	-	Auto	Manual
Model	*ROM [kB]*	*△ (%)*		
`toycar`	342.1	+1.2%	+2.2%	+1.4%
`resnet`	188.3	+8.0%	+14.1%	+8.5%
`vww`	420.5	+5.4%	+8.0%	+5.5%
`aww`	146.7	+15.6%	+23.1%	+15.7%

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
- Greatly improved performance from manually vectorizing depthwise convolutions
- Minimal additional ROM overhead from less aggressive loop unrolling

Future Work

- Support for sub-byte and mixed precision models
- Compile-time selection between kernels optimized for different targets
- Benchmarking of muRISCV-NN on open source RTL implementations of Zve32x
- Deployment of muRISCV-NN on commerical RISC-V Vector processors

This research is partially funded by the German Federal Ministry of Education and Research (BMBF) within the project Scale4Edge (grant number 16ME0127). The authors are responsible for the content of this publication.

SPONSORED BY THE Federal Ministry of Education and Research

Contact:
parker.jones@tum.de
philipp.van-kempen@tum.de
daniel.mueller-gritschneder@tuwien.ac.at

Open Source:
https://github.com/tum-ei-eda/