muRISCV-NN: Deep-Learning Inference Kernels for Embedded Platforms using the RISC-V Vector and Packed Extensions

Philipp van Kempen, Fabian Peddinghaus,
Jefferson Parker Jones, Rafael Stahl,
Daniel Müller-Gritschneider, Ulf Schlichtmann

Technical University of Munich
TUM School of Computation, Information and Technology
Chair for Electronic Design Automation

27.06.2023

This work has been developed in the ZuSE project Scale4Edge. Scale4Edge is funded by the German ministry of education and research (BMBF) (reference numbers: 16ME0122K-16ME0140+16ME0465). The authors are responsible for the content of this publication.
Agenda

1. **Motivation**
   - TinyML Workloads
   - TinyML Targets
   - TinyML Deployment
   - Edge ML Pipeline

2. **Methodology**
   - Implementation
   - Supported Operators
   - Infrastructure

3. **Evaluation**
   - MLPerf Tiny Benchmark
   - Performance Results
   - Overheads

4. **Conclusion**
   - Challenges
   - Outlook
Agenda

1. Motivation
   • TinyML Workloads
   • TinyML Targets
   • TinyML Deployment
   • Edge ML Pipeline

2. Methodology
   • Implementation
   • Supported Operators
   • Infrastructure

3. Evaluation
   • MLPerf Tiny Benchmark
   • Performance Results
   • Overheads

4. Conclusion
   • Challenges
   • Outlook
Motivation

- TinyML Workloads and targets are emerging rapidly
- RISC-V ISA becoming more popular
- Choosing the right ML Deployment Framework is non-trivial
TinyML Workloads

Deep AutoEncoder*  
Convolutional Neural Network (CNN)*

Transformer

Here:
- Quantized Networks only
- Inference only

*MLPerf™ Tiny workloads
TinyML Targets

Architectures for Embedded ML:

- CPU
- Vector
- Accelerators

Here:

- RISC-V MCU with optional SIMD support
  - Scalar: RV32IM[FD]C \(\rightarrow\) comparable to Cortex-M0
  - Packed: RV32IM[FD]CP \(\rightarrow\) comparable to Cortex-M4 (DSP)
  - Vector: RV32IM[FD]CV \(\rightarrow\) comparable to Cortex-M55 (DSP+MVEI/Helium)

RISC-V Packed Extension
- Still in development (v0.9.6)
- Sub-word SIMD (i8v4, i16v2,...)

RISC-V Vector Extension (RVV)
- Ratified in Nov. 2021 (v1.0)
- Super-word SIMD
- Embedded Vector Zve32x for MCUs
RISC-V Vector Cores

Academic
● Hwacha - UC Berkeley
● Ara - ETH Zurich
● Vitruvius+ - BSC
● RISC-V\(^2\) - Univ. of Thrace
● ...

Commercial
● OpenC906 / OpenC910 - Alibaba
● P270 / X280 - SiFive
● NX27V - Andes
● NS-72 / DR1000C - NSITEXE
● ...

Embedded Zve32x
● 2021 - Vicuna [1] - TU Wien
● 2022 - Spatz - ETH Zurich
● ...

---

TinyML Deployment

Frameworks

- TFLite for Microcontrollers [2] (TFLM)
  - Industry standard
  - Many vendor libraries
  - Hardcoded kernel → Generic or hand-tuned

- TVM [3]
  - Follows compiler-driven approach → Allows many optimizations & auto-tuning
  - Highly interesting research field
  - MicroTVM: Deployment of TVM programs to MCUs
  - Provides different ways to integrate accelerators: BYOC/UMA
Edge ML Pipeline

Model

TensorFlow Lite

Target

tvm

Program
Edge ML Pipeline

Target

Program
Edge ML Pipeline

**ARM:**
- Scalar
- DSP
- Helium

**RISC-V:**
- Scalar
- Packed
- Vector

Program
Edge ML Pipeline

**ARM:**
- Scalar
- DSP
- Helium

**RISC-V:**
- Scalar
- Packed
- Vector

---

**TensorFlow Lite**

**tvu**

---

**Generic C/C++**
Edge ML Pipeline

ARM:
- Scalar
- DSP
- Helium

RISC-V:
- Scalar
- Packed
- Vector

Provided by ARM
Edge ML Pipeline

ARM:
- Scalar
- DSP
- Helium

RISC-V:
- Scalar
- Packed
- Vector
Edge ML Pipeline

ARM:
- Scalar
- DSP
- Helium

RISC-V:
- Scalar
- Packed
- Vector

TensorFlow Lite

Generic C/C++
Agenda

1. Motivation
   • TinyML Workloads
   • TinyML Targets
   • TinyML Deployment
   • Edge ML Pipeline

2. Methodology
   • Implementation
   • Supported Operators
   • Infrastructure

3. Evaluation
   • MLPerf Tiny Benchmark
   • Performance Results
   • Overheads

4. Conclusion
   • Challenges
   • Outlook
Methodology

Design Idea
• Implement a kernel library for efficient NN inference on RISC-V MCUs

Approach
• Use ARM CMSIS-NN [4] as baseline implementation
• Keep interface consistent for compatibility with 3rd party tools

Implementation
• Supported modes:
  ▪ Scalar (portable): unchanged
  ▪ Vector (RVV): Implemented & hand-optimized
  ▪ Packed (RVP): Implemented & hand-optimized
• Using C-level intrinsic functions instead of Inline-Assembly!
Supported Operators & Types

**NN Operators:**
- (Depthwise Separable) Convolution
- Fully Connected
- Softmax
- ReLU
- Max Pooling / Average Pooling
- Elementwise Addition / Multiplication
- Singular Value Decomposition
- **New:** LSTM

**Data Types:**
- int8/int16 only

---

Only Quantized Networks → QNNs

Inference only → No on-device training
Infrastructure

Unit Tests:
• Passing the same unit tests as CMSIS-NN → bit exact!

Integration Tests:
• TFLite Micro (TFLM)
  ▪ Can be used as a drop-in replacement for CMSIS-NN by applying a minimal patch
• MicroTVM
  ▪ Reusing existing CMSIS-NN BYOC (Bring-Your-Own-Codegen) integration
• MLonMCU
  ▪ TinyML deployment tool used for benchmarking

CI/CD Flow:
• Code Style Checks
• Weekly builds + tests
• Automated Benchmarks

Miscellaneous:
• Providing instructions for obtaining toolchains (LLVM/GCC) and simulators
• Extensive user/developer documentation
Agenda

1. Motivation
   • TinyML Workloads
   • TinyML Targets
   • TinyML Deployment
   • Edge ML Pipeline

2. Methodology
   • Implementation
   • Supported Operators
   • Infrastructure

3. Evaluation
   • MLPerf Tiny Benchmark
   • Performance Results
   • Overheads

4. Conclusion
   • Challenges
   • Outlook
Benchmarks

**MLPerf Tiny Benchmarking Suite** [5]
Deep Learning Benchmarks for Embedded Devices

<table>
<thead>
<tr>
<th>Name</th>
<th>Use Case</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>aww</td>
<td>Keyword Spotting</td>
<td>DS-CNN</td>
</tr>
<tr>
<td>vww</td>
<td>Visual Wake Words</td>
<td>MobileNet</td>
</tr>
<tr>
<td>resnet</td>
<td>Image Classification</td>
<td>ResNet</td>
</tr>
<tr>
<td>toycar</td>
<td>Anomaly Detection</td>
<td>Deep AutoEncoder</td>
</tr>
</tbody>
</table>
Setup

Simulator
- Spike Instruction Set Simulator
- Support various RISC-V extensions, including V(ector) and P(acked)
- Also known as riscv-isa-sim

Toolchain
- LLVM 14: Scalar + Vector
- GCC (Draft): Packed

Flow
- All benchmarks generated with MLonMCU[6] Tiny Deployment tool
Results: muRISCV-NN vs. CMSIS-NN vs. TFLM

Default:
TFLM Reference Kernels
→ Not optimized

CMSIS-NN & muRISCV-NN:
Only ±10% differences
Results: Scalar vs. Vector vs. Packed

**V-Ext:**
Runtime falls with higher VLEN up to 2048 bits.

**P-Ext:**
Less overhead for configuration instructions and load/stores
Results: muRISCV-NN vs. TVM

Default:
NHWC Layout
Fallback Schedules

Optimized:
NCHW Layout → transformed
Tuned Schedules → > 2h per Model

Unoptimized: 2x speedup
Optimized: equivalent performance
Memory overheads

**ROM (constants + code size)**
- muRISCV-NN requires ~5% more ROM compared to TFLM reference implementation
- P-Ext kernel implementations require 5-10% more ROM

**RAM (intermediate buffers)**
- No extra overhead vs. TFLM default implementations
- Between 1.5-3.1 times less RAM compared to TVM kernels

Basically equivalent to CMSIS-NN memory metrics!
1. Motivation
   • TinyML Workloads
   • TinyML Targets
   • TinyML Deployment
   • Edge ML Pipeline

2. Methodology
   • Implementation
   • Supported Operators
   • Infrastructure

3. Evaluation
   • MLPerf Tiny Benchmark
   • Performance Results
   • Overheads

4. Conclusion
   • Challenges
   • Outlook
Challenges

Vector Extension (RVV)
• Can not utilize full vector length due to limited channel lengths in some convolutions
• Layout transformations not possible with CMSIS-NN/muRISCV-NN (NHWC only)

Packed Extension (RVP)
• Unable to achieve full potential due to TFLM quantization scheme (8bit + offset) → 16-bit inputs

Workarounds
• RISC-V does not support ARM rounding mode → Emulation is costly!

Maintenance
• Staying in-sync with upstream progress
Outlook

Work In Progress
• Optimization of kernels (specially P-Extension and special cases)
• Support more (academic) cores

Future Work
• Support hardware targets
• Alternative Rounding Modes
Efficient deep learning kernels for RISC-V microcontrollers.

- CMSIS-NN fork
  - Same unit tests
  - functionally equivalent!
- RISC-V extensions
  - P packed 0.9.6
  - V vector 1.0 (Zve32x)
- Supporting
  - ISS: Spike, riscvOVPsim, ETISS
  - RTL: Vicuna
  - HW: soon!

- Toolchains
  - RISC-V GCC & LLVM 14+
- Integration with
  - TensorFlow Lite
  - TVM
- Performance
  - Scalar Up to 10x faster than TFLM
  - On par with tuned TVM
  - Packed Up to 2.5x faster than Scalar
  - Vector Up to 11x faster than Scalar
Efficient deep learning kernels for RISC-V microcontrollers.

Open Source

https://github.com/tum-ei-eda/muriscv-nn
References


Questions
Bonus slides
Why ML at the Edge?

ML needs **Computing power & Memory bandwidth**

Offload computation to the **cloud**?

- ▲ Plenty of resources available
- ▼ High latency
- ▼ Low bandwidth
- ▼ Poor reliability
- ▼ Privacy concerns

➡ Cloud is not always a good choice!
Why Vectors? Why not SIMD?

**SIMD**
- SIMD has **fixed length** 😞
- Software **hard to maintain** 😞

**Vectors**
- Vector machines are **length-agnostic** 😊
- One software **fits all** 😊

![Diagram showing SIMD and Vectors operations]
Why Vectors? Why not SIMD?
RISC-V V Extension: ReLU Example

Reference in plain C

```c
for (uint16_t i = 0; i < size; i++) {
    if (data[i] < 0) data[i] = 0;
}
```

RISC-V Vector

```c
# a0=data, a1=size
for:
    vsetvli t0, a1, e8, m8, ta, ma    # Vectors of 8bit
    vle8.v v0, (a0)                    # Load bytes
    vmax.vx v0, v0, zero               # Apply activation
    vse8.v v0, (a0)                    # Store bytes
    add a0, a0, t0                      # Decrement size
    sub a1, a1, t0                      # Bump pointer
    bnez a1, for                        # Any more?
```
Copyright Notice

This presentation in this publication was presented as a tinyML® EMEA Innovation Forum. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org