AI/ML Inference on Low-cost Low-power multi-core RISC-V SoC

RT&I Team, Thales India, Bangalore

*Open Source project: No proprietary information present on this page.

We were able to deploy AI/ML inference for classifying and detecting various mechanical manufacturing faults, using Tensorflow-Lite on **multi-core RISCV-SoC**. **Custom Linux** with **TensorFlow-Lite** support was build using **Yocto** to run on multi-core RISC-V SoC, along with enabling Ethernet Link and webserver.

Part1: Hardware, Embedded-Linux and TensorFlow-Lite support

The PolarFire SoC Icicle kit is a low-cost multi-core development platform with Linux capable RISC-V core and FPGA fabric.

Linux was customized using Yocto, to have a small footprint, suitable for embedded applications. TensorFlow Lite Recipe was added to Yocto for enabling TensorFlow-lite inference in Python and C++. Various other library like numPy, pillow (python-imaging-library) were added to Linux build.

Ethernet was enabled on the embedded board for various web-services like getting data from various nodes/ip-cameras/cloud, upstreaming the results onto cloud and having real-time dashboard.

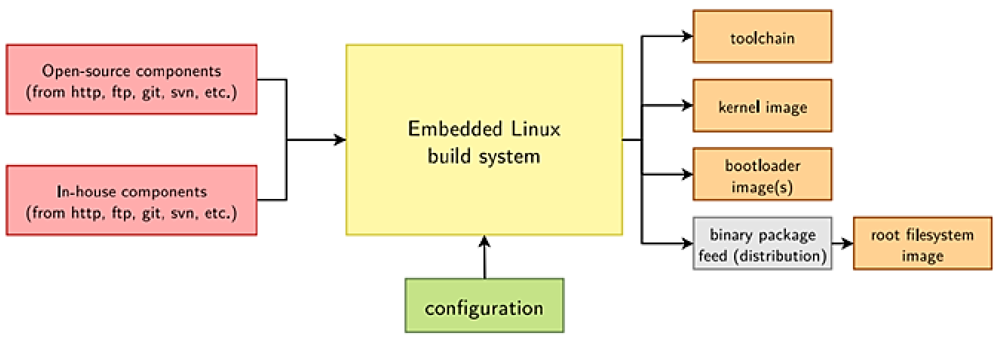
Open-source components (from http, ftp, git, svn, etc.)
In-house components (from http, ftp, git, svn, etc.)
Embedded Linux build system
configuration
toolchain
kernel image
bootloader image(s)
binary package feed (distribution)
root filesystem image

Fig1: Yocto Flow

meta-tensorflow lite

Yocto layer which enabled TensorFlow-Lite interpreter with Python / C++. All the dependencies Tensorflow-Lite source code was compiled using riscv-gnu-toolchain. Various dependencies for cmake, pybind11, pthread, libg etc which were natively compiled.

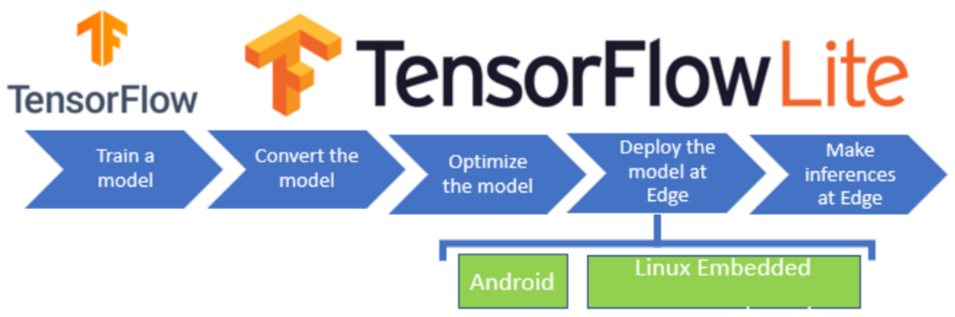

Fig2: TensorFlow-Lite steps

Execution flow for AI-inference would be:

1. Load the model using tflite-runtime-interpreter
2. Fetch the image
3. Allocate tensors/memory
4. Do pre-processing/crop/adjustment etc.
5. **Invoke the AI-model for inference**
6. Collect score
7. Annotate the result to assign/draw bounding-box
8. Post data onto webserver (through ethernet)

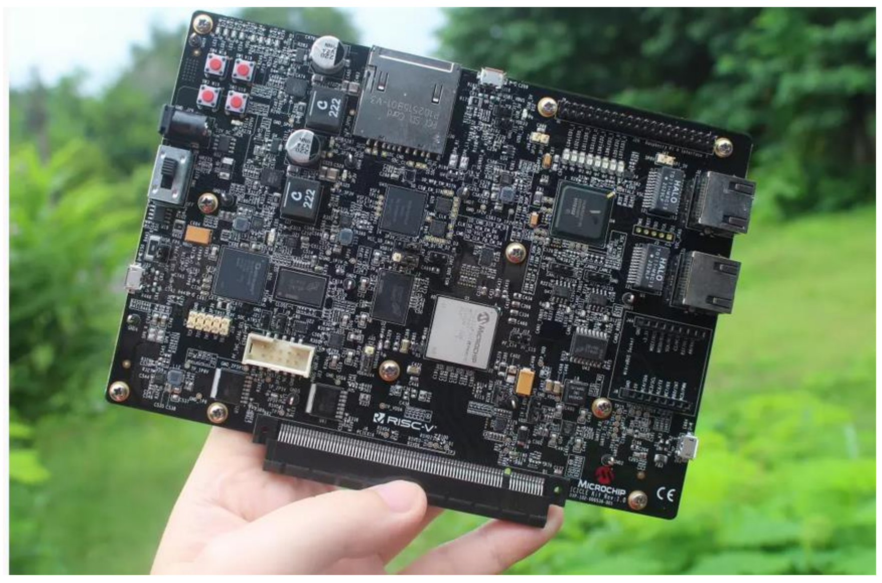

Fig3: PolarFire RISC-V Kit form-factor

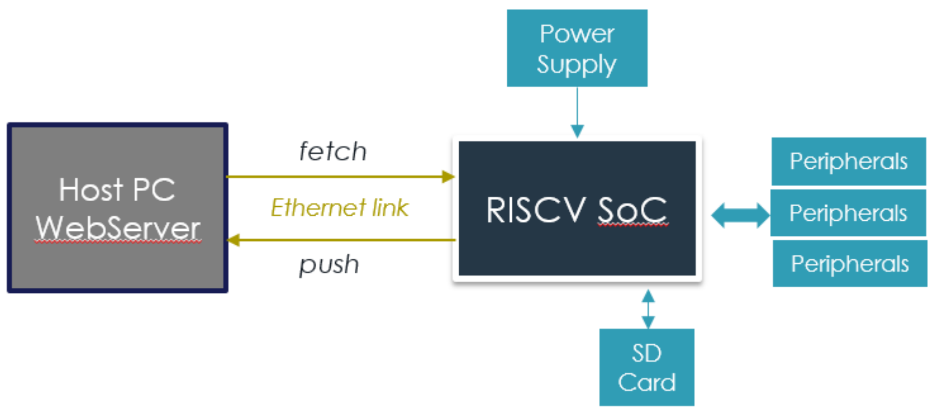

Fig4: Prototype setup diagram

This demonstration is in Python, while at-scale embedded solution should be preferable in C/C++.

Planned Vector eXtension support:

- We have evaluated Vector Coprocessors to pair with RISCV-core. This would help achieve higher throughput.
- Various performance characterization matrix-multiplication, convolution etc is done.
- llvm toolchain support is also planned for new instr.
- Also scoping out of having a Certifiable Compiler, for Aerospace and other critical application including Vector eXtension support.

Part 2: AI application flow

Details on AI flow like various neural-network training strategy and methodology, converting model to tflite, performing image-classification and detection to find out various mechanical manufacturing defect etc.