TFLu & CMSIS-NN: Efficient Neural Network kernels for Arm Cortex-M CPUs

Felix Johnny & Fredrik Knutsson
Maintainer, CMSIS-NN
Feb 2nd, 2020
CMSIS

Pathway to the Arm eco-system

- Cortex Microcontroller Software Interface Standard
- Consistent, generic, and standardized software building blocks
- Available for all Cortex-M and Cortex-A5, A7, A9 processors
- Open source – public development on GitHub: https://github.com/ARM-software/CMSIS_5

6,000+ devices supported with CMSIS

Used in many projects
> 1,200,000 source files public on GitHub

Device family packs
> 3,000,000 pack downloads in past 6 months

© 2019 Arm Limited
CMSIS 5
Consistent software framework for Arm Cortex-M and Cortex-A5/A7/A9 based systems

Application code

CMSIS-Pack
CMSIS-RTOS
Real-time execution
CMSIS-NN
Machine learning
CMSIS-DSP
Signal processing
CMSIS-Driver
Middleware interface
CMSIS-CORE
Processor core and peripheral access
Peripheral HAL
Device specific
CMSIS-SVD
Peripheral description
CMSIS-DAP
Debug access
CMSIS-Zone
System Partitioning

Debugger

Arm® Cortex® processor
Communication peripherals
Specialized peripherals
CoreSight™ debug logic
Access Filter (MPU, SAU)

System-on-chip
Why target Arm Cortex-M CPUs?

• Accelerate deployment of Machine Learning on edge devices

• Enable Machine Learning on more than 47 billion shipped units

• Networks with lower memory footprint and MACs are available now

- MobileNet V2
  • 3.38 MB parameters
  • ~307 million MACs

- Person Detect (TFL)
  • 250 kBytes
  • ~7 million MACs
CMSIS-NN : The Highlights

• Optimized library for Neural Network operators to best utilize the capabilities of Arm Cortex-M CPUs.

• Optimized for speed and memory utilization.

• Utilization of SIMD capability is a major part of the optimizations.

• Follows int8 symmetric quantization for in, out and weight tensors
Registers for SIMD operations

DSP Extension vs Arm Helium™ technology (M-Profile Vector Extension)

DSP – 32 bit general purpose (GP) registers

- 8 bit layout => 4 operands
- 16 bit layout => 2 operands
- 13 General Purpose registers

MVE – 128 bit vector (Q) registers

- 8 bit layout => 16 operands/lanes
- 16 bit layout => 8 operands/lanes
- 32 bit layout => 4 operands/lanes
- 64 bit layout => 2 operands/lanes
- 13 GP registers and eight 128 bit Q registers
Use Case: Multiply Accumulate

Core loop comparison of MAC operation

DSP Extension

- SIMD MAC instruction operates on two data element pairs.
- Handles even number of elements.

```
.LBB1_2: @ %for.body
    ldr    r12, [r0], #4
    ldr    r4, [r1], #4
    smlad  r2, r12, r4, r2
    le     lr, .LBB1_2
```

M-Profile Vector Extension

- SIMD MAC instruction can operate on sixteen data element pairs.
- Tail predication enables handling odd and even number of elements in the same loop.
- Accumulation done on General purpose registers

```
.LBB0_2: @ %for.body
    vldrb.u8 q0, [r0], #16
    vldrb.u8 q1, [r1], #16
    vmlava.s8 r12, q0, q1
    letp    lr, .LBB0_2
```
CMSIS-NN & TensorFlow Lite for Microcontrollers
Access to optimized kernels through TFLu

- Collaboration with TFL micro team
- Optimized kernels enabled by simply using “TAGS” in the TFLu build system
- CMSIS-NN “glue” is here: /tensorflow/lite/micro/kernels/cmsis-nn
- Automatic fallback to reference kernels when optimization is not applicable/available
Build TFLu with CMSIS-NN, simple example

- Step 1: Clone TensorFlow repository from GitHub
  ```
git clone https://github.com/tensorflow/tensorflow
  ```

- Step 2: There is an experimental int8 model for person detection available. Compile it with optimized kernels
  ```
make -f tensorflow/lite/micro/tools/make/Makefile TAGS=cmsis-nn TARGET=<your M4-or-later target>
person_detection_int8
  ```
Build TFLu with CMSIS-NN, continued...

- What happens now?

  - CMSIS repo downloaded
  - CMSIS-NN kernels selected when applicable
  - Binary ready to run
  - Third party code download
  - CMSIS repo downloaded
  - CMSIS-NN kernels selected when applicable
  - Binary ready to run
  - Third party code download
  - CMSIS repo downloaded
  - CMSIS-NN kernels selected when applicable
  - Binary ready to run
  - Third party code download
  - CMSIS repo downloaded
  - CMSIS-NN kernels selected when applicable
  - Binary ready to run
  - Third party code download
  - CMSIS repo downloaded
  - CMSIS-NN kernels selected when applicable
  - Binary ready to run
What more

• More info
/tensorflow/lite/micro/kernels/cmsis-nn/README.md

• Profiling your network – use the network tester!
/tensorflow/lite/micro/examples/network_tester/README.md
Why use Mbed Enabled boards?

- Low threshold to prototype projects on Cortex-M CPUs
- Models and kernel tests from TFL micro can be run on Mbed Enabled boards
Useful links

- CMSIS GitHub: https://github.com/ARM-software/CMSIS_5

- Supported operators in CMSIS: https://github.com/ARM-software/CMSIS_5/tree/develop/CMSIS/NN

- Person Detection Int8 example (under development): https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/micro/examples/person_detection_experimental

- Mbed supported boards: https://os.mbed.com/platforms