Running Machine Learning on Arm’s Ethos-U55 NPU
Welcome!

- Tweet us: @ArmSoftwareDev -> #AIVTT

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: developer.arm.com/techtalks
Our upcoming Arm AI Tech Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>November 2nd</td>
<td>Getting started with running Machine Learning on Arm Ethos-U55</td>
<td>Arm</td>
</tr>
<tr>
<td>November 16th</td>
<td>Hands-on workshop with the Arm ML Embedded Evaluation kit for Ethos-U55</td>
<td>Arm</td>
</tr>
<tr>
<td>November 30th</td>
<td>Getting started with Arm NN on Android, in just 5 minutes</td>
<td>Arm</td>
</tr>
<tr>
<td>December 14th</td>
<td>Improve PyTorch App Performance with Android NNAPI Support</td>
<td>Arm</td>
</tr>
</tbody>
</table>

Visit: developer.arm.com/techtalks
Presenter

• Software engineer in Arm’s Machine Learning team
• Develop ML applications on Arm silicon
• Previously, part of Arm’s IoT team

George Gekov
Does anybody enjoy their ML software running slowly?
Agenda

- What is the Arm Ethos-U55 microNPU?
- What software stack to use on the Ethos-U55?
- How to optimise a neural network?
- Demo!
Ethos-U55: First microNPU for Cortex-M CPUs

Example: Typical ML Workload for a Voice Assistant

- **Speed to inference**
  - Cortex-M7
  - Cortex-M55
  - Cortex-M55 + Ethos-U55
  - 50x
  - 6x

- **Energy efficiency**
  - Cortex-M7
  - Cortex-M55
  - Cortex-M55 + Ethos-U55
  - 25x
  - 7x

- Faster responses
- Smaller form-factors
- Improved accuracy

Latency and energy spent for all tasks listed combined: voice activity detection, noise cancellation, two-mic beamforming, echo cancellation, equalizing, mixing, keyword spotting, OPUS decode, and automatic speech recognition.
Develop for the Arm Ethos-U55 without a development board!

How to create software applications when NPU silicon is not commercially available yet?

**Arm Virtual Hardware**

- Fixed Virtual Platform (FVP) – digital twin of a development board with Ethos-U55 & Cortex-M55
- Corstone-300(sse-300), available as part of Arm Virtual Hardware
- MAC = Multiply Accumulate
  - Ethos-U55 supports 32, 64, 128, 256 MACs

**Arm Cortex-M55 and Arm Ethos-U55**
What’s Arm Virtual Hardware?
Virtual Hardware Targets are the IoT equivalent of Virtual Machines

- An Arm Virtual Hardware Target is a functionally accurate representation of a physical SoC, simulating its software-visible behavior
- Runs as a simple application in a Linux environment for easy scalability in the cloud
- Remove dependency from RTL or silicon availability
- Available as a public beta for multiple configurations of the Arm Corstone-300 subsystem, incorporating the Cortex-M55 CPU and Ethos-U55 uNPU.

[Diagram showing layers of IoT application and Virtual Hardware with Host OS (Linux) and Cloud infrastructure]

www.arm.com/virtual-hardware
ML embedded evaluation kit
Open-source, Apache 2.0

- [https://review.mlplatform.org/plugins/gitiles/ml/ethos-u/ml-embedded-evaluation-kit](https://review.mlplatform.org/plugins/gitiles/ml/ethos-u/ml-embedded-evaluation-kit)

- Ready to use applications for Arm Ethos-U55
Why use the ML embedded evaluation kit?

Three main benefits

Performance evaluation

• Number of NPU cycles
• Amount of memory transactions

Software stack evaluation

- Keyword spotting
- Image classification
- Visual Wake Word
- ASR
- Anomaly detection

Custom workflow

• Test custom NN performance on the Ethos-U55
• Framework to implement new ML use-cases

Inference Runner
## Software stack evaluation

### ML use cases

<table>
<thead>
<tr>
<th>Application common</th>
<th>TensorFlow Lite for Microcontrollers</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAL</td>
<td>Arm Ethos-U55 driver</td>
</tr>
<tr>
<td>x86-64 stubs</td>
<td>Arm Cortex-M BSP</td>
</tr>
<tr>
<td>x86 hardware</td>
<td>Arm Cortex-M55 &amp; Arm Ethos-U55 hardware</td>
</tr>
</tbody>
</table>
TFLµ Operator Support – CMSIS-NN and Ethos-U NPU

- **Supported operators**
  - Abs, Add, Average_Pool_2D, Concatenation, Conv_2D, Depthwise_Conv_2D, Fully_Connected, Leaky_ReLu, Logistic, Maximum, Max_Pool_2D, Minimum, Mul, Pack, Quantize, ReLu, ReLu6, ReLu_N1_to_1, Reshape, Resize_Bilinear, Slice, SoftMax, Split, Split_V, Squeeze, Strided_Slice, Sub, TanH, Transpose_Conv, Unpack and others. See SUPPORTED_OPS.md (generated from vela)

- **Optimized operators**
  - The library has a roadmap of Quarterly releases to expand scope and improve performance

- **Fallback to reference kernels**

---

**Start**

- Micro TensorFlow Lite runtime
- CMSIS-NN optimized operators
- Ethos microNPU driver
- Optimized custom operators for the microNPU

**Offline Optimization**

- TensorFlow Lite .TF Input File
- Modified .TF Input File

**30+ operators**

**80+ operators**

© 2021 Arm
Vela compiler

- Open source python tool: https://review.mlplatform.org/admin/repos/ml/ethos-u/ethos-u-vela
- Pypi: https://pypi.org/project/ethos-u-vela/
  
  pip3 install ethos-u-vela
- Top level functionality:
  - Parses a model
  - Optimises the graph
  - Tensor allocation
  - Command stream generation
  - Saves optimised model
- Configurable behaviour: https://review.mlplatform.org/plugins/gitiles/ml/ethos-u/ethos-u-vela/+/refs/heads/master/OPTIONS.md
- Supported ops: https://review.mlplatform.org/plugins/gitiles/ml/ethos-u/ethos-u-vela/+/refs/heads/master/SUPPORTED_OPS.md
Vela workflow

Initial model

Vela configuration

Call Vela

Optimised model

$ vela mobilenet_v2_1.0_224_INT8.tflite --accelerator config=ethos-u55-128 --optimise Performance --config vela.ini --memory-mode=Shared_Sram --system config=Ethos_U55_High_End_Embedded

- Input: tflite file & vela configuration
- Output: tflite file
- Input model:
  - Can run on CPU (with CMSIS kernels if possible),
  - Cannot run on microNPU
- Output model:
  - "Ethos-u" op cannot run on CPU but can run on microNPU
  - All fallback ops run on CPU (with CMSIS kernels if possible)
Vela configuration

What needs to be configured?

- Memory latencies and bandwidths (Deeply embedded, high-end, ..)
- microNPU configuration (32, 64, 128, 256 MACs)
- Memory mode
- Example configuration file:
  - https://review.mlplatform.org/plugins/gitiles/ml/ethos-u/ethos-u-vela/+/refs/heads/master/vela.ini

![Diagram]

1. Input model
2. System configuration
3. Vela
4. Output model

Example configuration:

```ini
; System Configuration
; Ethos-U55 High-End Embedded: SRAM (4 GB/s) and Flash (8.5 GB/s)
(System_Config,Ethos_U55_High_End_Embedded)
core_clock=500us
axi0_port-Sram
axi1_port-OffChipFlash
Sram_clock_scale=1.0
Sram_burst_length=32
Sram_read_latency=32
Sram_write_latency=32
OffChipFlash_clock_scale=9.125
OffChipFlash_burst_length=128
OffChipFlash_read_latency=64
OffChipFlash_write_latency=64
```
Run one of the available applications on the Ethos-U55 microNPU

Quick way to run an application & how to do a non-default build

- For a default build – use build_default.py script

- For a non-default build

  1. Specify Vela configuration and compile the model
  2. Configure the build system with CMake
  3. Compile the project with make
  4. Run the application binary on FVP or FPGA
What is cycle accurate & what is not cycle accurate?

Fixed Virtual Platform (Arm Virtual Hardware)

- Arm Ethos-U55: cycle approximate
- Arm Cortex-M55: functionally accurate

MPS3 FPGA

- Arm Ethos-U55: cycle accurate
- Arm Cortex-M55: cycle accurate
arm

Demo time
Summary

• What is the Arm Ethos-U55 microNPU?
• What software stack to use?
• How can you optimise a neural network for the Arm Ethos-U55 microNPU?
• How can you run an application on the Arm Ethos-U55 microNPU?
Try it yourself!

• Download the source code

• Try running an application yourself

• If you have a custom neural network, try running it on the Ethos-U55 and tell us how you get on https://discuss.mlplatform.org/c/ml-embedded-evaluation-kit/

• Access Arm Virtual Hardware (AVH) on AWS marketplace as Amazon Machine Image – www.arm.com/virtual-hardware
  • Attend AI Tech Talk on Nov 16th for hands-on workshop with AVH
  • 100hrs of free AWS EC2 CPU credits for first 1,000 qualified users