“microTVM: a Tensor Compiler for Bare Metal”

Andrew Reusch - OctoML

[Japan Area Group] – January 19, 2021
tinyML Talks Sponsors

Arm

EDGE IMPULSE

maxim integrated

Qeexos

SynSense

Additional Sponsorships available – contact Bette@tinyML.org for info
Arm: The Software and Hardware Foundation for tinyML

1. Connect to high-level frameworks
   - Profiling and debugging tooling such as Arm Keil MDK

2. Supported by end-to-end tooling
   - Optimized models for embedded
     - Runtime (e.g. TensorFlow Lite Micro)
     - Optimized low-level NN libraries (i.e. CMSIS-NN)

3. Connect to Runtime
   - RTOS such as Mbed OS
   - Arm Cortex-M CPUs and microNPUs

AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers
@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with little to no drop in accuracy for inference on resource-limited MCUs

Reduce model optimization trial & error from weeks to days using Deeplite’s design space exploration

Deploy more models to your device without sacrificing performance or battery life with our easy-to-use software

BECOME BETA USER bit.ly/testdeeplite
TinyML for all developers

- Acquire valuable training data securely
- Enrich data and train ML algorithms
- Test impulse with real-time device data flows
- Test
- Embedded and edge compute deployment options
- Real sensors in real time
- Open source SDK
- Arduino library
- WebAssembly
- C++ library

Get your free account at [http://edgeimpulse.com](http://edgeimpulse.com)
Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

The biggest (3MB flash and 1MB SRAM) and the smallest (256KB flash and 96KB SRAM) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels.

The new MAX78000 implements AI inferences at over 100x lower energy than other embedded options. Now the edge can see and hear like never before.
Qeexo AutoML for Embedded AI
Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

Key Features

- Wide range of ML methods: GBM, XGBoost, Random Forest, Logistic Regression, Decision Tree, SVM, CNN, RNN, CRNN, ANN, Local Outlier Factor, and Isolation Forest
- Easy-to-use interface for labeling, recording, validating, and visualizing time-series sensor data
- On-device inference optimized for low latency, low power consumption, and a small memory footprint
- Supports Arm® Cortex™- M0 to M4 class MCUs
- Automates complex and labor-intensive processes of a typical ML workflow – no coding or ML expertise required!

Target Markets/Applications

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

For a limited time, sign up to use Qeexo AutoML at automl.qeexo.com for FREE to bring intelligence to your devices!
Reality AI Tools® software

- Automated Feature Exploration and Model Generation
- Bill-of-Materials Optimization
- Automated Data Assessment
- Edge AI / TinyML code for the smallest MCUs

Reality AI solutions

- Automotive sound recognition & localization
- Indoor/outdoor sound event recognition
- RealityCheck™ voice anti-spoofing

https://reality.ai  info@reality.ai  @SensorAI  Reality AI
SynSense builds ultra-low-power (sub-mW) sensing and inference hardware for embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
## Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, February 2</td>
<td><strong>Martino Sorbaro</strong>&lt;br&gt;R&amp;D Scientist, SynSense</td>
<td>Always-on visual classification below 1 mW with spiking convolutional networks on Dynap™-CNN</td>
</tr>
</tbody>
</table>

Webcast start time is 8 am Pacific time

Please contact [talks@tinyml.org](mailto:talks@tinyml.org) if you are interested in presenting
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums  youtube.com/tinyml

Please use the Q&A window for your questions
Andrew Reusch is a Software Engineer at OctoML and a core contributor to the microTVM project. Prior to OctoML, he worked on digital IC design and embedded firmware for medical devices at Verily, an Alphabet company. Andrew holds a Bachelor of Engineering in Computer Engineering from the University of Washington.
microTVM: a Tensor Compiler for Bare Metal
TinyML Meetup - Tokyo

Andrew Reusch
Outline

- What is μTVM?
- How μTVM Works
- Demo Walkthrough
- Future Directions
- Q&A
What is $\mu$TVM?

using TVM to run models on bare-metal devices

libmodel.a
...Bare Metal?

To μTVM, bare metal is not just:

🚀 Raspberry Pi
  - These (usually) have operating systems

🚀 Reserved Cloud Instances
  - These still have Virtual Memory

🚀 Running outside a VM
  - Again, still a traditional OS
...Bare Metal?

Bare metal is (often) IoT-class devices

- AzureSphere
- Arduino
- Cortex-M class micro-controllers
μTVM works in places without...

🚫 Operating Systems
- no files, DLLs, .so, memory mapping, kernels

🚫 Virtual Memory
- No malloc, C++ RAII, exceptions, ...

🚫 Advanced Programming Languages
- No Rust or Python required…
  (But we like those and you could use them!)
The deployment challenge
The deployment challenge

frameworks

- PyTorch
- Caffe2
- TensorFlow
- mxnet
- ONNX
- TensorFlow Lite
The deployment challenge
The deployment challenge

<table>
<thead>
<tr>
<th>PyTorch</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Caffe2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TensorFlow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mxnet</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ONNX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TensorFlow Lite</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### The deployment challenge

<table>
<thead>
<tr>
<th></th>
<th>PyTorch</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>
The deployment challenge
The deployment challenge
Bare Metal Deployment Challenges

- Less abstraction than full OS
  - Less tools to work with

- Resources are tighter
  - Scheduling is harder

- Demands are unique per-chip and per-project
  - Code reuse is tricky
TVM: Bridging the gap as a DL compiler and runtime

- Cut capital and operational ML costs
- Build your model once, run anywhere
- Reduce model time-to-market

Open source, optimization framework for deep learning.

ML-based Optimizations

Backends for x86, nVidia/CUDA, AMD, ARM, MIPS, RISC-V, etc
TVM is an emerging industry standard

Every “Alexa” wake-up today across all devices uses a model optimized with TVM

“[TVM enabled] real-time on mobile CPUs for free...We are excited about the performance TVM achieves.” More than 85x speed-up for speech recognition model.

Bing query understanding: 112ms (Tensorflow) -> 34ms (TVM). QnA bot: 73ms->28ms (CPU), 10.1ms->5.5ms (GPU)

“TVM is key to ML Access on Hexagon” - Jeff Gehlhaar, VP Technology

Unified ML compilation stack for CPU, GPU, NPU built with TVM
The μTVM Approach

● Batteries Included
  ○ μTVM can be used with only the standard C library

● Compute-centric
  ○ μTVM does not configure the SoC--it only runs computations
  ○ μTVM integrates with RTOS like Zephyr and mBED for SoC configuration

● Transparent
  ○ μTVM binaries can be compiled directly from source
int32_t fused_conv2d_right_shift_add() {
  // ...
}
How μTVM Works

```c
int main() {
    // configure SoC
    TVMInitializeRuntime();
    TVMGraphRuntime_Run();
}
```

```c
int32_t fused_conv2d_right_shift_add() {
    // ...
}
```
Working with the TVM Compiler

Model import

Relay Module

```python
# [version = "0.0.5"]
def main(%data : Tensor[(1, 3, 64, 64), int8],
       %weight : Tensor[(8, 3, 5, 5), int8]) {
  %1 = nn.conv2d(
    %data,
    %weight,
    padding=[2, 2],
    channels=8,
    kernel_size=[5, 5],
    data_layout="NCHW",
    kernel_layout="OIHW",
    out_dtype="int32");
  %3 = right_shift(%1, 9);
  %4 = cast(%3, dtype="int8");
  %4
}
Model import

Optimize Operators (AutoTVM)

 Relay Module

TensorIR

# version = "0.0.5"
def @main(%data : Tensor[(1, 3, 64, 64), int8],
%weight : Tensor[(8, 3, 5, 5), int8]) {
    %1 = nn.conv2d(
        %data,
        %weight,
        padding=[2, 2],
        channels=8,
        kernel_size=[5, 5],
        data_layout="NCHW",
        kernel_layout="OIHW",
        out_dtype="int32";
    %3 = right_shift(%1, 9);
    %4 = cast(%3, dtype="int8";
    %4
}

primfn(placeholder_2: handle,
    placeholder_3: handle,
    T_cast_1: handle) -> ()
allocate(kernel_vec, int8, [600]) {
    for (bs.c.fused.h.fused: int32, 0, 64) "parallel" {
        for (w: int32, 0, 64) {
            for (vc: int32, 0, 3) {
                data_vec[(((bs.c.fused.h.fused*192) +
                    (w*3)) + vc]) =
                    (uint8*)placeholder_5[(((vc*4096) +
                        (bs.c.fused.h.fused*64)) + w)]
            }
        }
    // ...
Working with the TVM Compiler

Model import

Optimize Operators
(AutoTVM)

Generate C/LLVM library

Relay Module

TensorIR

C Source Code

OctoML
Working with the TVM Compiler

Model import

Parameters or Weights

Optimize Operators (AutoTVM)

Generate C/LLVM library

Simplified Parameters

Compiled Operators (.c)

Operator Call Graph (JSON)
Running the model end-to-end

Graph JSON

Compiled Operators

Simplified Parameters

FuncRegistry

Graph Runtime

conv2d_input

conv2d

Intermediate

bias_add

output

p1

p2
How μTVM Works

```c
int main() {
    // configure SoC
    TVMInitializeRuntime();
    TVMGraphRuntime_Run();
}

int32_t fused_conv2d_right_shift_add() {
    // ...
}
```
Putting the pieces together

TVM Compiler Outputs

- Simplified Parameters
- Compiled Operators
- Graph JSON

Graph Runtime
- Library, from TVM

RPC Client/Server
- Library, from TVM

Model Inputs & Outputs (RAM)
- Data, from user

main()
- SoC config & startup libraries, from RTOS/vendor
Putting the pieces together - host-driven

Graph JSON

Graph Runtime

RPC Client

Simplified Parameters (FLASH)

Model Inputs & Outputs (RAM)

Compiled Operators

RPC Server

UART, Semihosting, USB, etc

main()
Putting the pieces together - standalone
microTVM Reference Virtual Machine

- Lots of moving pieces...
  - Physical hardware
  - TVM compiler
  - GCC, LLVM, etc
  - RTOS (Zephyr, mBED), library code
  - SoC configuration / main()

- How can we collaborate?
  - Use a “Reference VM” to freeze as much of the software as possible
  - Attach hardware to VM with USB passthrough
  - See MicroTVM Reference VM Tutorial for more
Demo Walkthrough
Future Directions
μTVM in 2020

- Experimental μTVM: April
- Blog post + roadmap: June
- Standalone μTVM: Dec
const DLTensor weights = {1, 2, ...};
const DLTensor biases = {4, 2, 7, ...};

int32_t classifier(DLTensor* input,
                  DLTensor* output) {
    DLTensor* intermediate =
        TVMBackendAllocWorkspace(512);
    conv2d(input, &weights, intermediate);
    bias_add(intermediate, &biases, output);
    TVMBackendFreeWorkspace(intermediate);
    return rv;
}
Next for μTVM: Hardware-aware Quantization

- Data-aware quantization v2
  - Allows quantizing more networks from within TVM

- Ultra-low-bit-width quantization
  - Could reduce the overall model memory footprint

- See HAGO PR and Ziheng Jiang’s talk “Hardware-aware Quantization in TVM” on Dec 4

Riptide: Fast End-to-End Binarized Neural Networks. MLSys 2020 (March 3rd)

OctoML
Next for μTVM: Heterogeneous Execution

- Hardware acceleration offers:
  - Lower power
  - Better performance
  - More parallelism

- Potential TVM improvements:
  - CRT multi-context execution
  - TIR subgraph offloading

See ARM’s lighting talk:
“Ethos-U55: microNPU Support for μTVM” by Manupa Karunaratne
Next for μTVM: Memory Planning

- Current memory planner has limitations:
  - Unaware of device memory layout
  - Requires a heap-based memory allocator

- New directions:
  - Tensor pinning
  - Accelerator-aware planning
  - Bring-your-own memory planning
Next for μTVM: Increased Coverage

- Much of the work so far has been infrastructure-focused

- Next step: increase μTVM coverage in terms of:
  - Supported ISA
  - Optimized Model Operators

- Auto-Scheduling can help
Next for μTVM: Developer Experience

- Create “getting started” experience
  - Generate e.g. Arduino, Zephyr, etc projects

- Improve TVM C runtime
  - Handle faults and report through RPC server
  - Gather runtime stats to increase visibility on-device
  - Support more complex runtime scenarios -- sensing, multitasking, etc.

- Documentation
  - Targeted to developers from multiple backgrounds -- ML, firmware, etc.
  - Add design documentation and more tutorials
Getting Involved

- TVM has a vibrant open-source community of 460+ contributors
- Contributions are welcomed:
  - The microTVM M2 Roadmap details larger upcoming projects. The community submits RFCs (often w/ PoC) to discuss implementation.
  - PRs for bugfixes, small enhancements, documentation changes are always welcomed!
- Questions? Proposals? RFCs?

Please post on our Discourse forum: https://discuss.tvm.ai
Q&A

- Code at https://github.com/areusch/microtvm-blogpost-eval

- Tutorials at https://tvm.apache.org/docs/tutorials/index.html#micro-tvm
This presentation in this publication was presented as a tinyML® Talks webcast. 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