tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"microTVM: a Tensor Compiler for Bare Metal"

Andrew Reusch - OctoML

[Japan Area Group] – January 19, 2021

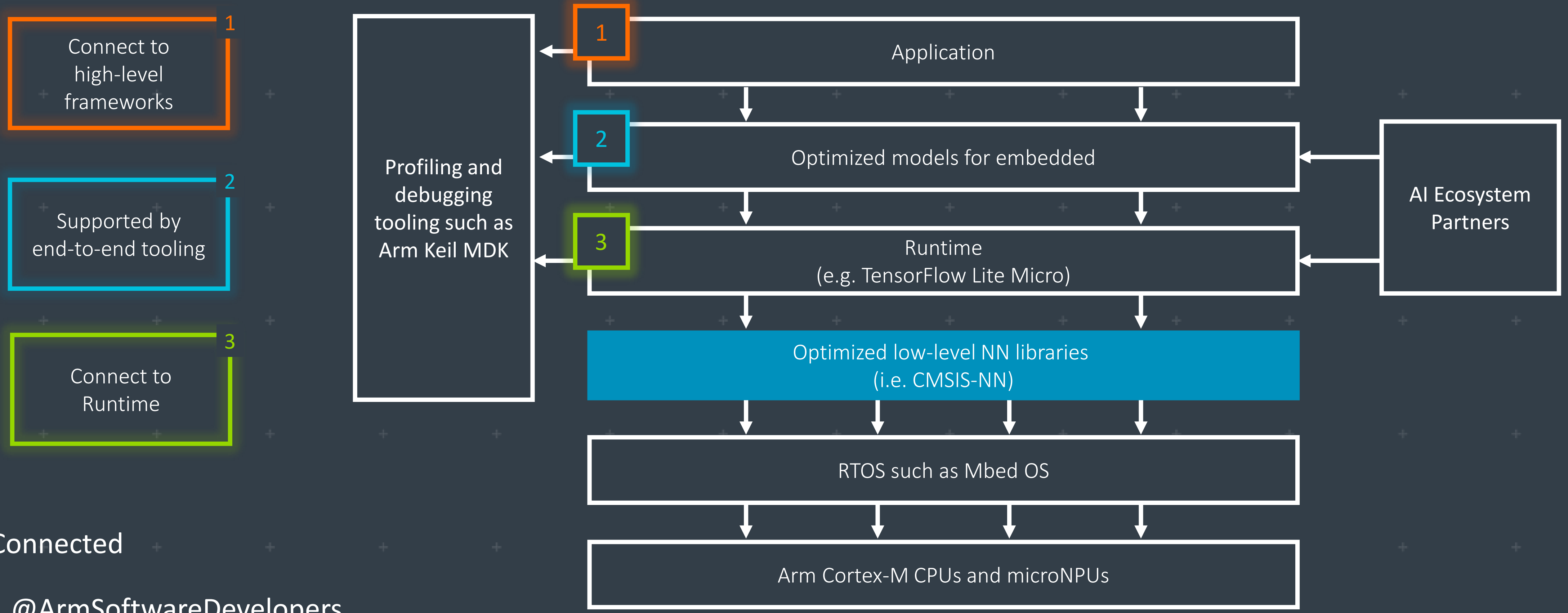
TINY ML

tinyML Talks Sponsors

tinyML Strategic Partner

Additional Sponsorships available – contact Bette@tinyML.org for info

Arm: The Software and Hardware Foundation for tinyML



Stay Connected

@ArmSoftwareDevelopers

@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm

arm

Deeplite

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**

APPLY NOW

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

BECOME BETA USER **bit.ly/testdeeplite**

mobilityXlab arm AI 100 CBINSIGHTS

TinyML for all developers

C++ library

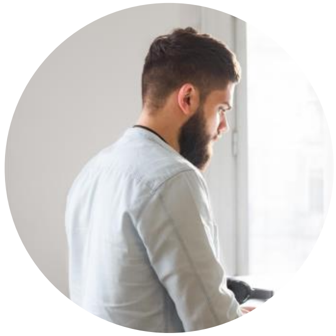
Arduino library

WebAssembly

Dataset

Acquire valuable training data securely

Enrich data and train ML algorithms



Edge Device

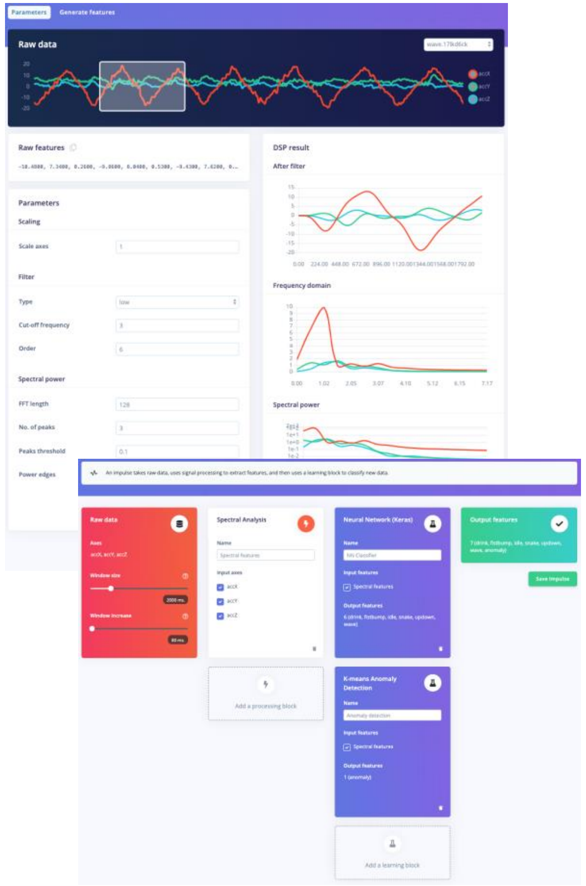
Real sensors in real time
Open source SDK

Impulse

Embedded and edge compute deployment options

Test impulse with real-time device data flows

Test



Get your free account at http://edgeimpulse.com

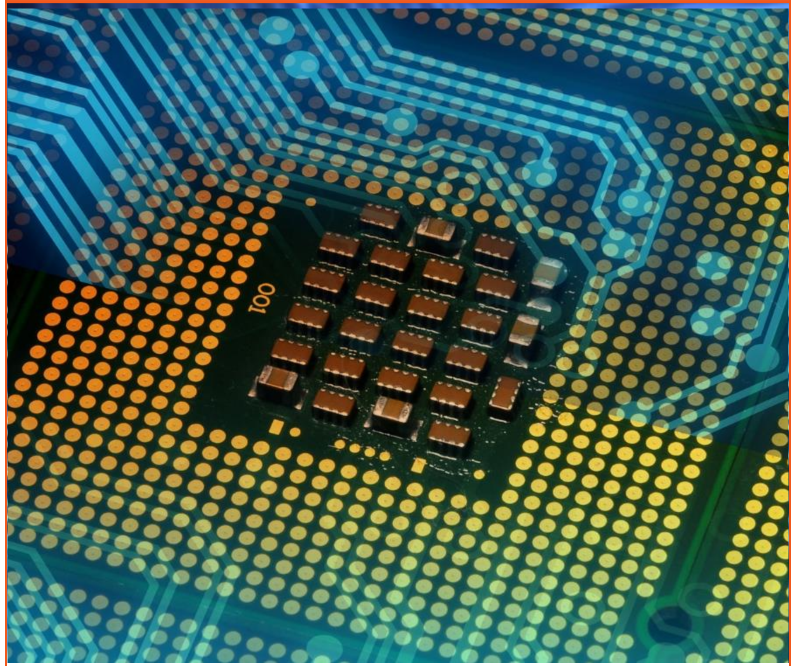
Maxim Integrated: Enabling Edge Intelligence

www.maximintegrated.com/ai

Sensors and Signal Conditioning

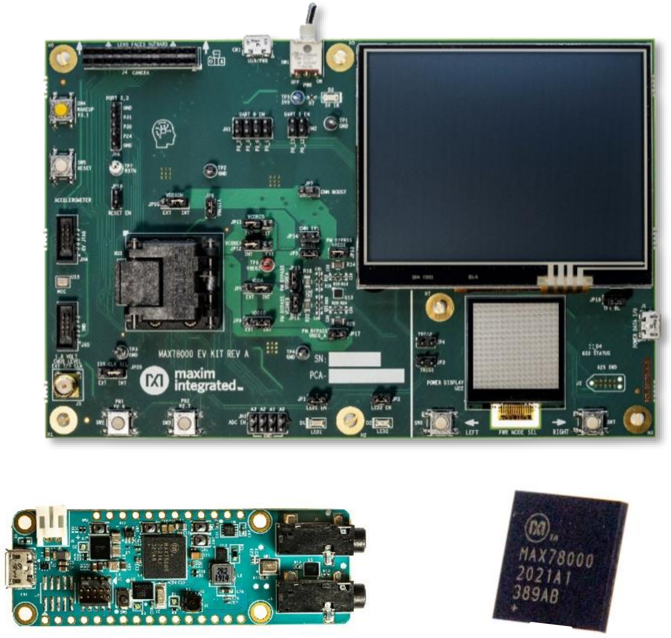
Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals.

Low Power Cortex M4 Micros



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

Advanced AI Acceleration



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

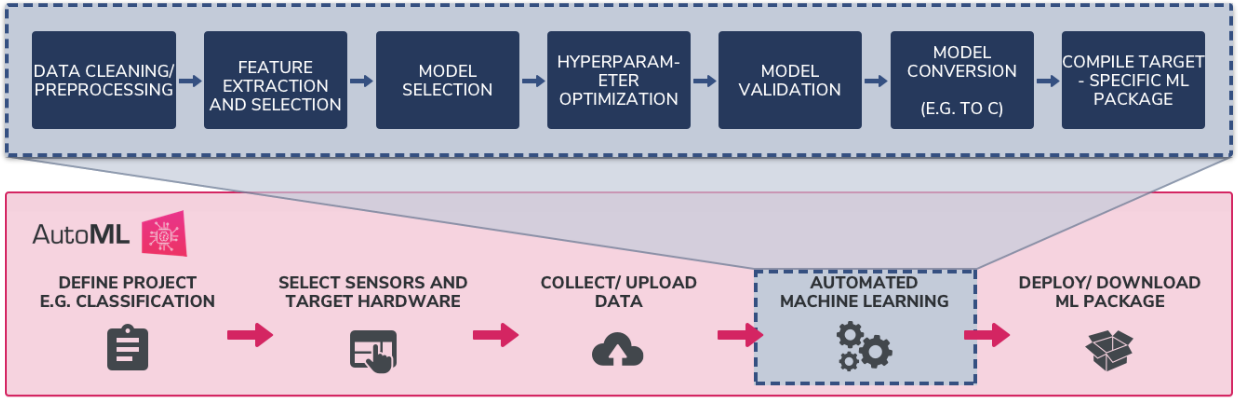
Key Features

- 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

QEEXO AUTOML: END-TO-END MACHINE LEARNING PLATFORM



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

Reality AI®

is for

building products

https://reality.ai info@reality.ai @SensorAI Reality AI

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

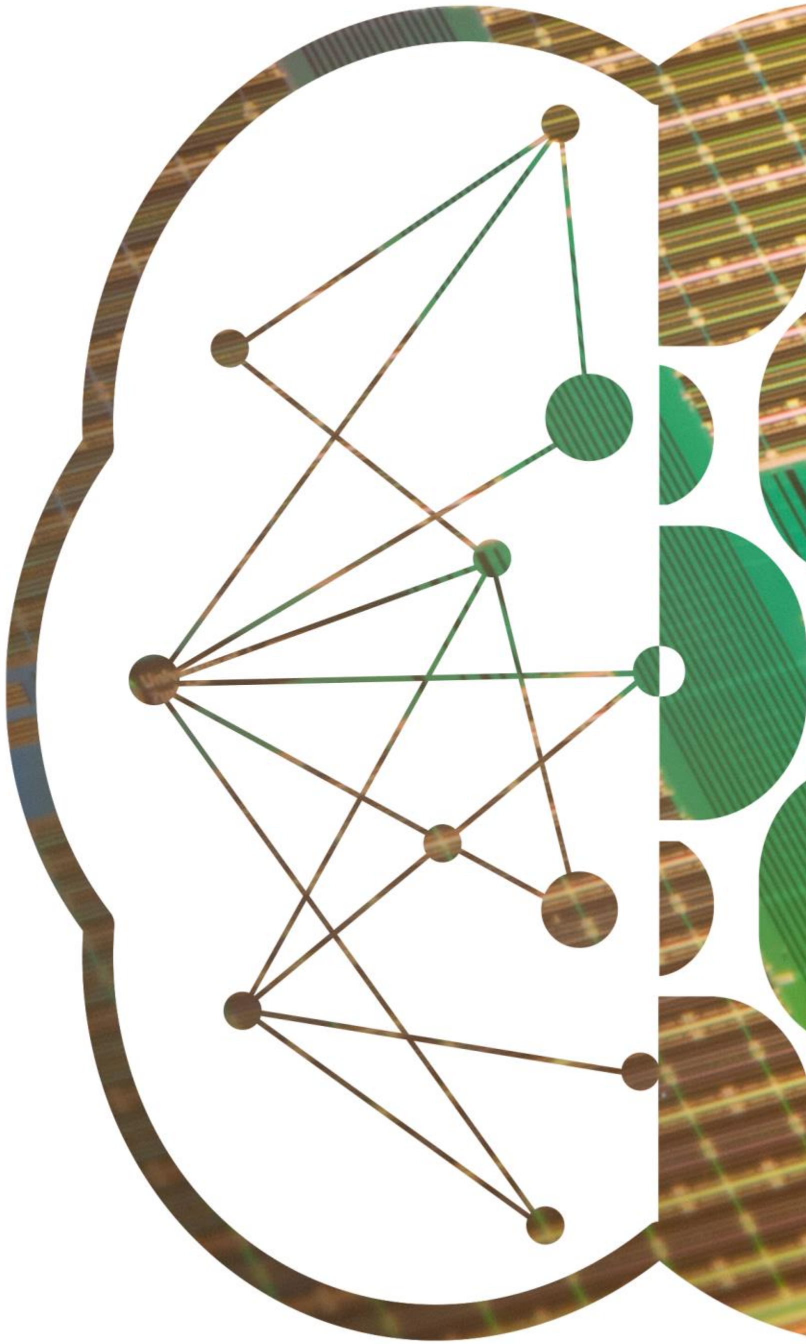
Indoor/outdoor sound event recognition

RealityCheck™ voice anti-spoofing

SynSense

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

Date	Presenter	Topic / Title
Tuesday, February 2	**Martino Sorbaro** R&D Scientist, SynSense	Always-on visual classification below 1 mW with spiking convolutional networks on Dynap™-CNN

Webcast start time is 8 am Pacific time

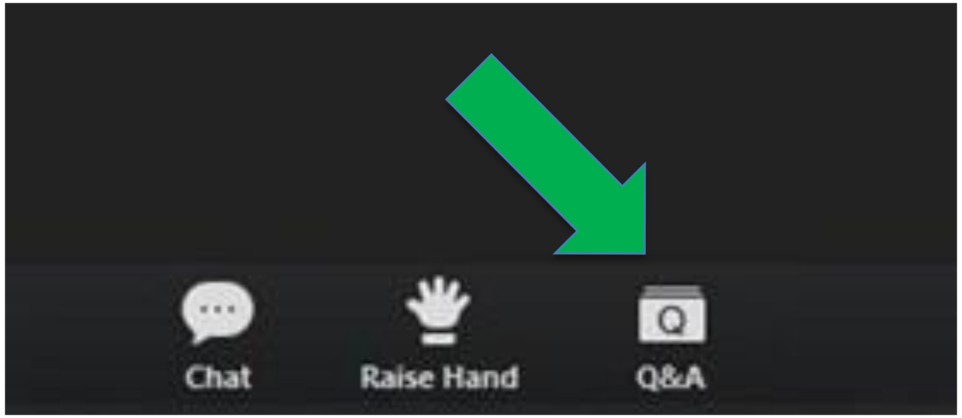
Please contact talks@tinyml.org if you are interested in presenting

Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums youtube.com/tinyml

Please use the Q&A window for your questions



Andrew Reusch

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

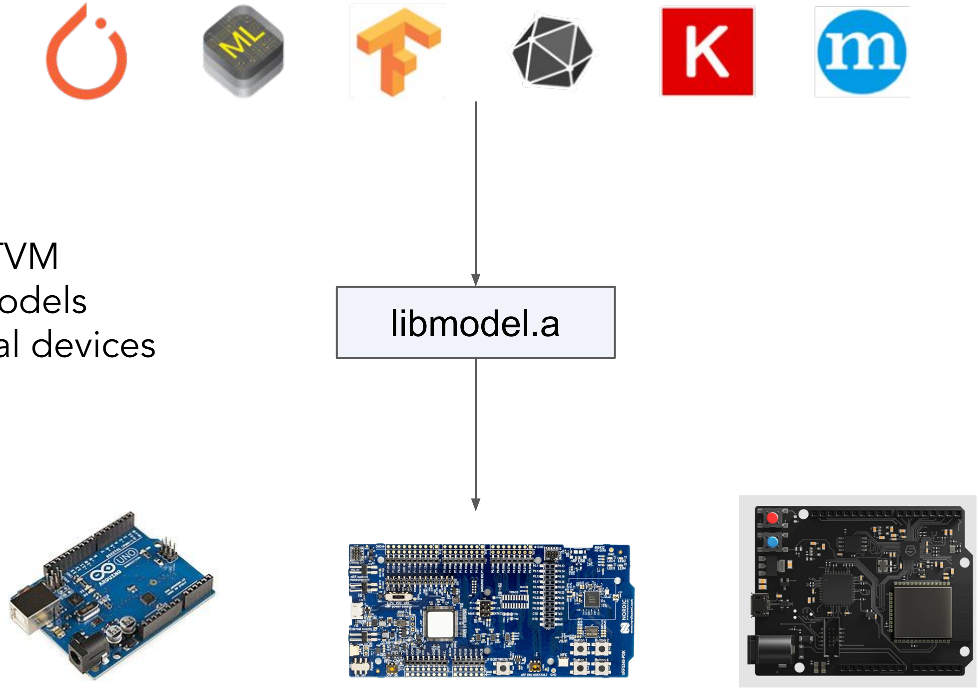
Andrew Reusch

Outline

- What is µTVM?
- How µTVM Works
- Demo Walkthrough
- Future Directions
- Q&A

What is µTVM?

using TVM
to run models
on bare-metal devices



…Bare Metal?

To µTVM, bare metal is not just:

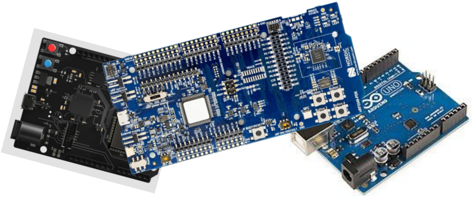
- 🚀 Raspberry Pi
  - These (usually) have operating systems
- 🚀 Reserved Cloud Instances
  - These still have Virtual Memory
- 🚀 Running outside a VM
  - Again, still a traditional OS

...Bare Metal?

Bare metal is (often) IoT-class devices

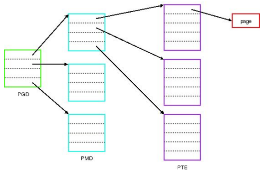
- ⚡ AzureSphere
- ⚡ Arduino
- ⚡ Cortex-M class micro-controllers

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

PyTorch

Caffe2

TensorFlow

mxnet

ONNX

TensorFlow Lite

The deployment challenge

frameworks

PyTorch

Caffe2

TensorFlow

mxnet

ONNX

TensorFlow Lite

The deployment challenge

targets →

PyTorch

Caffe2

TensorFlow

mxnet

ONNX

TensorFlow Lite

The deployment challenge

targets →

PyTorch	?	?	?	?	?	?	?	?	?	?
Caffe2	?	?	?	?	?	?	?	?	?	?
TensorFlow	?	?	?	?	?	?	?	?	?	?
mxnet	?	?	?	?	?	?	?	?	?	?
ONNX	?	?	?	?	?	?	?	?	?	?
TensorFlow Lite	?	?	?	?	?	?	?	?	?	?

The deployment challenge

targets →

PyTorch	✓	?	?	✓	?	?	?	?	?	?
Caffe2	✓	?	?	✓	?	?	?	?	?	?
TensorFlow	✓	?	?	✓	?	?	?	?	?	?
mxnet	✓	?	?	✓	?	?	?	?	?	?
ONNX	✓	?	?	✓	?	?	?	?	?	?
TensorFlow Lite	✓	?	?	✓	?	?	?	?	?	?

The deployment challenge

PyTorch	?	?	?	?	?	?	?	?	?	?
Caffe2	?	?	?	?	?	?	?	?	?	?
TensorFlow	?	?	?	?	?	?	?	?	?	?
mxnet	?	?	?	?	?	?	?	?	?	?
ONNX	?	?	?	?	?	?	?	?	?	?
TensorFlow Lite	?	✓	?	?	?	?	?	?	?	?

The deployment challenge

PyTorch	?	?	?	?	?	?	?	?	?	?
Caffe2	?	✗	?	?	?	?	?	?	?	?
TensorFlow	?	?	?	?	?	?	?	?	?	?
mxnet	?	?	?	?	?	?	?	?	?	?
ONNX	?	?	?	?	?	?	?	?	?	?
TensorFlow Lite	?	?	?	?	?	?	?	?	?	?

Bare Metal Deployment Challenges

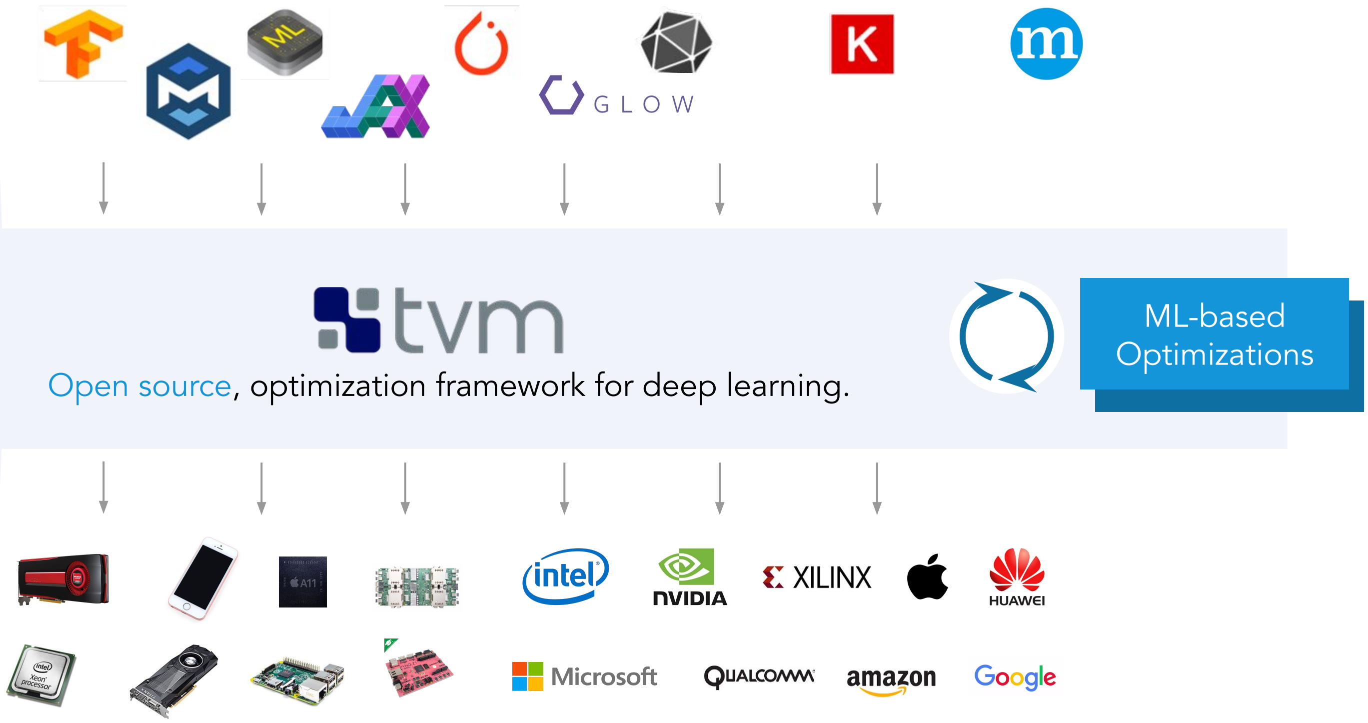
- Less abstraction than full OS
  - Less tools to work with
- Resources are tighter
  - Scheduling is harder
- Demands are unique per-chip and per-project
  - Code reuse is tricky

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



Backends for
x86, nVidia/CUDA, AMD, ARM, MIPS, RISC-V, etc

TVM is an emerging industry standard

aws — Every "Alexa" wake-up today across all devices uses a model optimized with TVM

tvm / The Apache Software Foundation — Open source ~460 contributors from industry and academia.

Facebook — "[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.

TVM CONF × 2020 — 950+ attendees

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

Qualcomm — "TVM is key to ML Access on Hexagon" - Jeff Gehlhaar, VP Technology

arm — Unified ML compilation stack for CPU, GPU, NPU built with TVM

OctoML AMD XILINX SiMa.ai UNTETHER AI

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

How µTVM Works

```
int32_t fused_conv2d_right_shift_add() {
  // ...
}
```

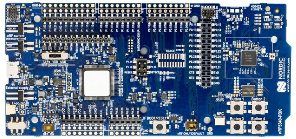
How µTVM Works

tvm

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

+

```
int32_t fused_conv2d_right_shift_add() {
  // ...
}
```



Working with the TVM Compiler

Model import

Relay Module

```
#[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
}
```

Working with the TVM Compiler

Model import

Optimize Operators (AutoTVM)

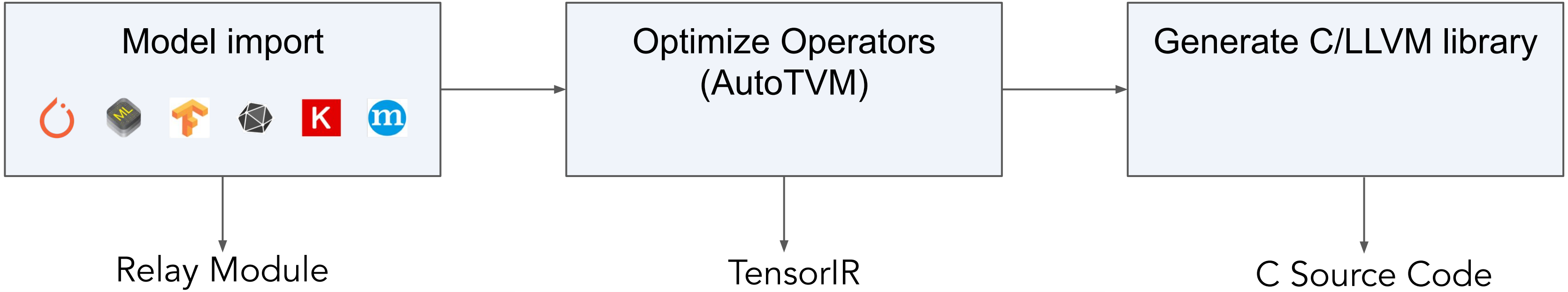
Relay Module

```
#[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
}
```

TensorIR

```
 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



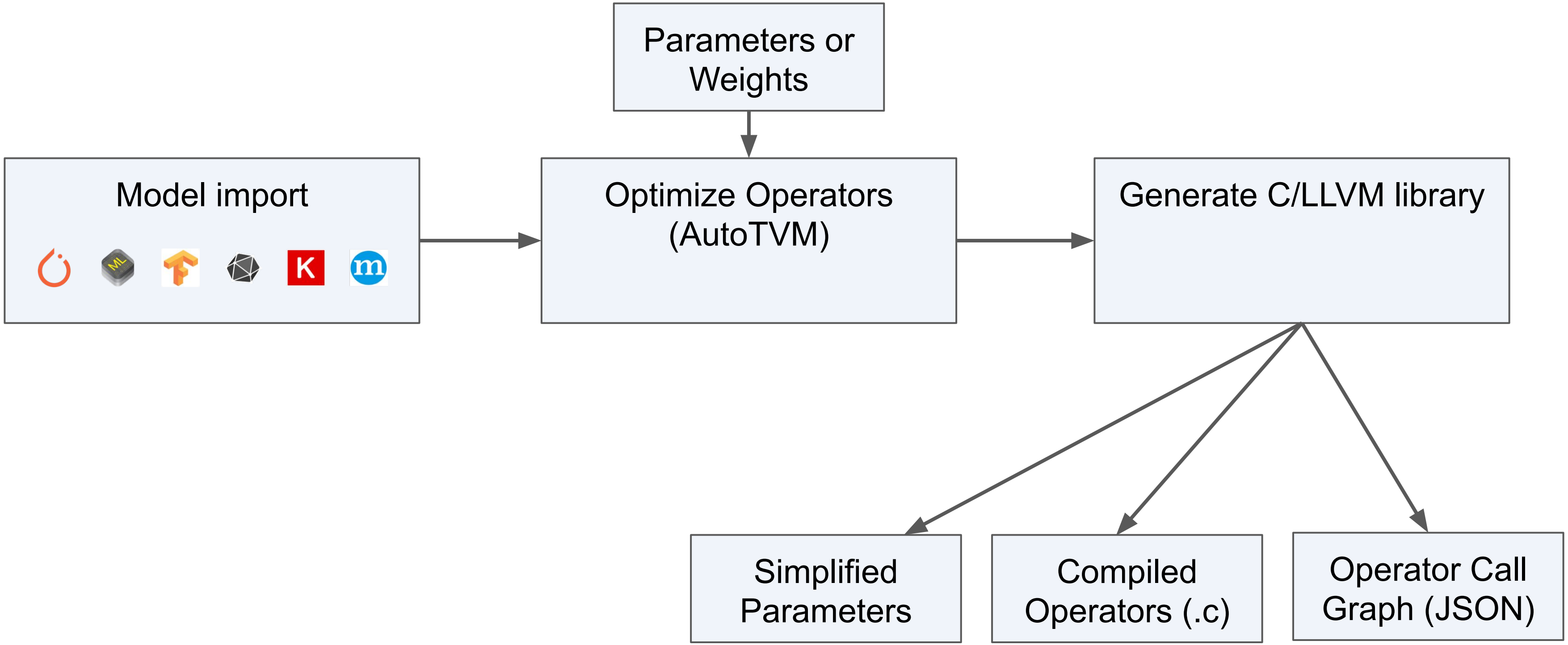

```
#[version = "0.0.5"]
def @main(%data : Tensor[(1, 3, 64, 64), int8],
          %weight : Tensor[(8, 3, 5, 5), int8]) {
    %1 = nn.conv2d(
         %data,
         %weight,
         padding=[2, 2],
         channels=8,
         kernel_size=[5, 5],
         data_layout="NCHW",
         kernel_layout="OIHW",
         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)]
        }
      }
    }
    // ...
```

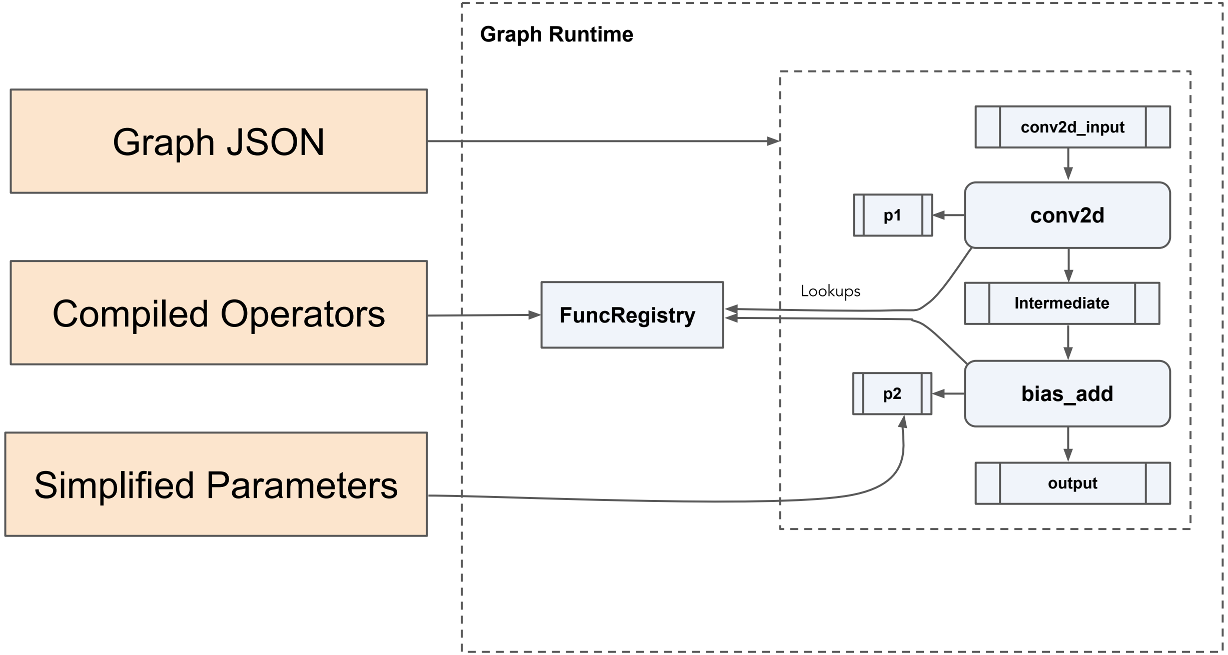
```
int32_t
fused_nn_contrib_conv2d_NCHWc_right_shift_cast(
void* args, void* arg_type_ids,
int32_t num_args, void* out_ret_value,
void* out_ret_tcode, void* resource_handle) {
  void* data_pad = TVMBackendAllocWorkspace(1,
dev_id, (uint64_t)13872, 1, 8);
  for (int32_t i0_i1_fused_i2_fused = 0;
i0_i1_fused_i2_fused < 68;
++i0_i1_fused_i2_fused) {
    for (int32_t i3 = 0; i3 < 68; ++i3) {
      for (int32_t i4 = 0; i4 < 3; ++i4) {

((uint8_t*)data_pad)[((((i0_i1_fused_i2_fused *
204) + (i3 * 3)) + i4))] = (((((2 <=
i0_i1_fused_i2_fused) && (i0_i1_fused_i2_fused
< 66)) && (2 <= i3)) && (i3 < 66)) ?
((uint8_t*)placeholder)[(((((i0_i1_fused_i2_fus
ed * 192) + (i3 * 3)) + i4) - 390))] :
(uint8_t)0);
```

OctoML

Working with the TVM Compiler



Running the model end-to-end



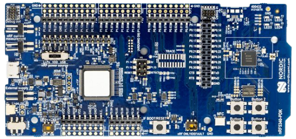
How µTVM Works

tvm

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

+

```
int32_t fused_conv2d_right_shift_add() {
  // ...
}
```



TVM Compiler Output

Putting the pieces together

TVM Compiler Outputs

Simplified Parameters

Compiled Operators

Graph JSON

Graph Runtime

Library, from TVM

RPC Client/Server

Library, from TVM

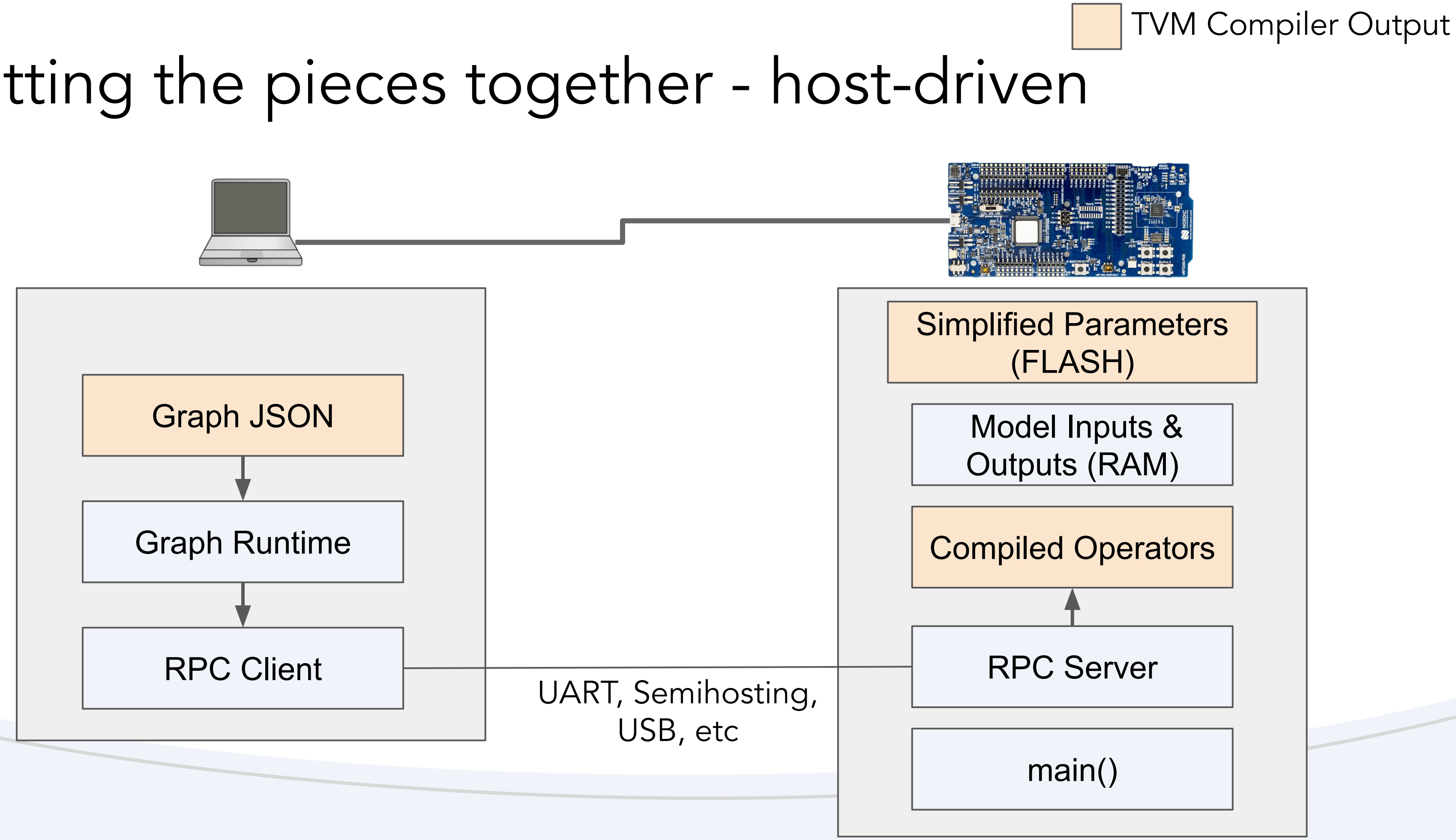
Model Inputs & Outputs (RAM)

Data, from user

main()

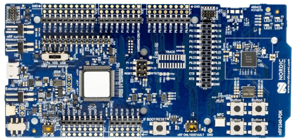
SoC config & startup libraries, from RTOS/vendor

Putting the pieces together - host-driven



TVM Compiler Output

Putting the pieces together - standalone



Simplified Parameters (FLASH)

Inputs (RAM)

Compiled Operators

Graph Runtime

Graph JSON

main()

microTVM Reference Virtual Machine

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

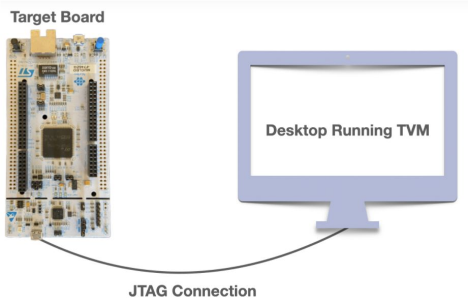
- How can we collaborate?
    - Use a “Reference VM” to freeze as much of the software as possible
    - Attach hardware to VM with USB passthrough
    - See MicroTVM Reference VM Tutorial for more

Demo Walkthrough

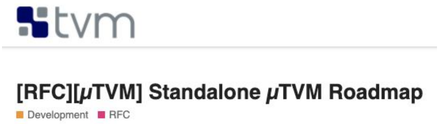
OctoML

Future Directions

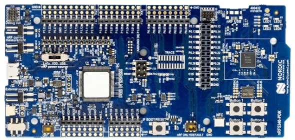
μTVM in 2020



Experimental μTVM



Blog post + roadmap



Standalone μTVM

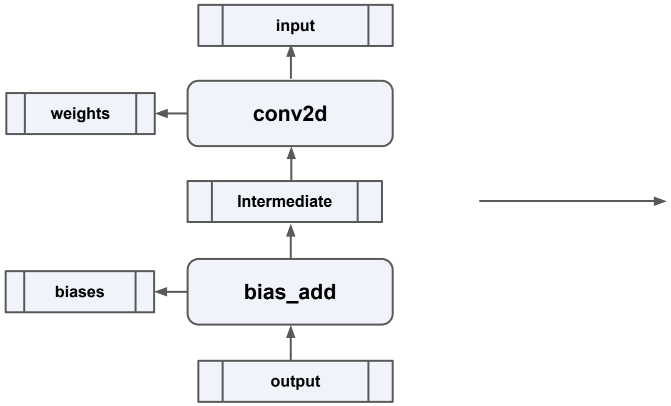
April

June

Dec

OctoML

Next for µTVM: Ahead-of-Time Compiler



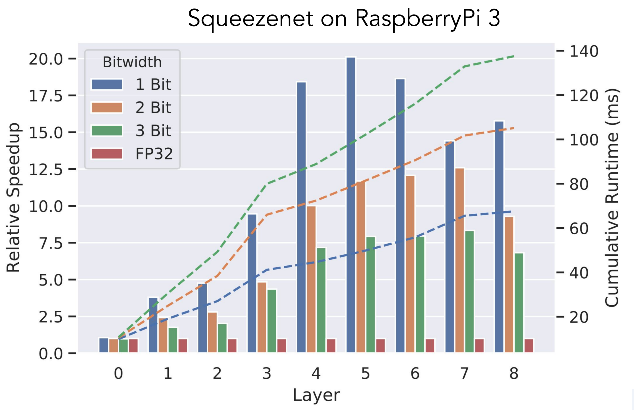

```
const DLTensor weights = {1, 2, ...};
const DLTensor biases = {4, 2, 7, ...};

int32_t classifier(DLTensor* input,
                   DLTensor* output) {
  DLTensor* intermediate =
     TVMBackendAllocWorkspace(512);

  conv2d(input, &weights, intermediate);
  bias_add(intermediate, &biases, output);

  TVMBackendFreeWorkspace(intermediate);
  return rv;
}
```

NOTE: this is a sample from the AOT compiler in development--expect API changes and an RFC.

OctoML

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)

Joshua Fromm · Meghan Cowan · Matthai Philipose · Luis Ceze · Shwetak Patel

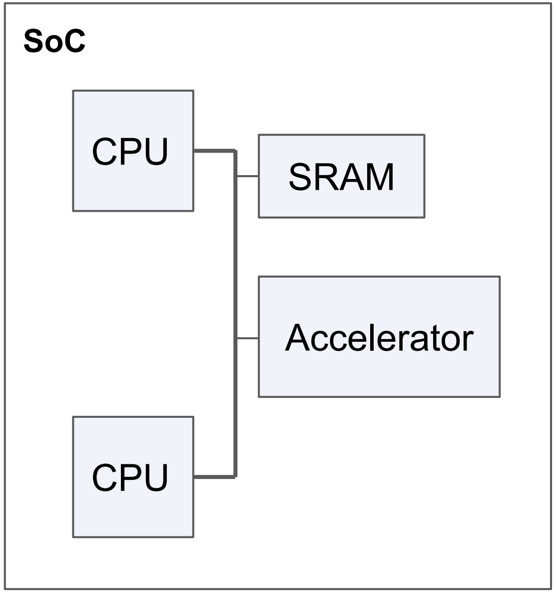
OctoML

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 uTVM"
  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

Copyright Notice