tinyML Summit

Miniature dreams can come true...

March 28-30, 2022 | San Francisco Bay Area

www.tinyML.org
Compiling TinyML Models with microTVM

Andrew Reusch
Software Engineer, OctoML
TVM Project Management Committee Member
AI at the Edge
AI at the Edge
AI at the Edge
The Deployment Challenge

This route may be easy:

- A fast impl may exist for each layer
- Tools exist to translate the model graph into C
- Robust test and debug tools
The Deployment Challenge

This route may be difficult:

- Hand-optimize each layer
- Tune memory usage of your impl
- Test and debug on your own

where the model runs
We need to speak a common language

```python
x = keras.Conv2d(2, 3, activation='relu')
k NatalMaxPool2d(x)
```
Model Deployment is a Workflow

Proprietary dataset

Source Model

Re-training

Cloud model

Quantization?

Lightweight model

Optimize for Target

Accurate model

Validation

Sensor data

Deployable Model

Integrate into Firmware

Firmware Binary

Program Device

Programmed Device

Validation in-situ

TinyML!
The Many Languages of TinyML Deployment

Lots of different expertise needed from different roles:

- Model developers
- MLSys engineers
- Firmware engineers
- Test engineers
- And more?
Apache TVM: Bridging the gap as a DL compiler and runtime

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
What is microTVM?

using TVM to run models on bare-metal devices
microTVM works in places without...

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

⚠️ Virtual Memory
- No malloc, C++ RAII, exceptions, …

⚠️ Advanced Programming Languages
- No C++, Rust, Python, …
  (But we like those and you could use them!)
The microTVM Vision

> Codify and Automate Model Deployment onto any hardware platform

Including:

- Tailoring the model to fit
- Deciding how to leverage the hardware available
- Optimizing the model for application and platform
- Running, debugging, validating the model end-to-end on-device

```python
x = keras.Conv2d(2, 3, activation='relu')
k keras.MaxPool2d(x)
```

```c
int32_t tvmgen_model_run(
    struct tvmgen_default_input
    struct tvmgen_default_output
...`
Languages of the TVM Compiler

Model import

Relay Module

Optimize Operators

(TAutoTVM)

Generate C/LLVM library

TIR Module

C Source Code or LLVM IR

```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";
}
```

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

```c
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)]
  }
}
  // ...
```
Why TIR?

- Separates implementation from definition
  - Relay is functional, TIR is procedural

- TVM can automate implementation of layers
  - AutoTVM and MetaScheduler allows TVM to automatically tile, reorder, loop-unroll, etc
  - Future: Auto-tensorization allows TVM to generate an implementation given knowledge of vector intrinsics

- Define procedural optimizations one level above codegen
  - Example: common subexpr elimination could be used with CPU, GPU, or even with expression that span both

- Graph-level modeling in TIR
  - Memory allocations, buffer copies, and operator order can all be determined ahead-of-time
Accelerating microTVM by hand

- Hand-tuned implementations can complement TVM’s optimization strategies

- Software libraries
  - CMSIS-NN for microTVM
  - Other examples: TensorRT, CuBLAS, CUTLASS, etc.

- Hardware accelerators
  - Ethos-U for ARM platforms
microTVM Deployment: Today

$ tvmc compile mymodel.tflite
  --target="c ..."
  --output-format=mlf

Model Library Format (.tar)

{“model_name”: “mobilen”,
 “memory”: {
  ...
 },
}

Machine-readable metadata

def @main(data: Tensor[%
  %0 = conv2d(%data, %we

Model source for TVM (Relay)

(multiple formats possible)

Machine-readable parameters

int32_t tvmgen_resnet_c
  void* data = ((DLTens

Implemented model (.c or .o)
microTVM Deployment: Today

```python
def @main(data: Tensor[
  %0 = conv2d(%data,%we
```

$ tvmc micro generate-project -t arduino mymodel.tflite

Machine-readable parameters

implemented model (.c or .o)

Model source for TVM (Relay)

(multiple formats possible)

Machine-readable metadata

```json
{"model_name": "mobilene
"memory": {
  ...
},
}
```
microTVM as a Workflow

- **Source Model**
- **Proprietary dataset**
- **Re-training**
- **Cloud model**
- **Quantization?**
- **Lightweight model**
- **Sensor data**
- **Optimize for Target**
- **Accurate model**
- **Validation**
- **Deployable Model**
- **Integrate into Firmware**
- **Firmware Binary**
- **Program Device**
- **Programmed Device**
- **Validation in-situ**
- **TinyML!**
Recently landed work in microTVM

- Support for automatic optimization via AutoTVM
- Whole-program memory planning: Unified Static Memory Planner
- Automatically create Arduino projects from an arbitrary model
Upcoming work in microTVM

- **UMA - Universal Modular Accelerator infrastructure**
  - API specifically designed to guide the addition of accelerator-specific compilation flows

- **C Device API** – first-class support for accelerators in the microTVM runtime

- Improved CLI support for AutoTVM using tvmc

- tvm (and microTVM) coming to PyPI
  - pip install apache-tvm
Getting Involved

- microTVM is a community-driven effort, and we welcome new contributions!

- See the microTVM Roadmap for a sense of planned work you can get involved with

- We welcome new ideas and questions on our Discuss forum and Discord, and new contributors to join our weekly TVM Community Meeting

https://tvm.apache.org/community
By the way, we’re hiring!

OctoML started in July 2019 - $130M in venture capital, now 100+ people and growing!

We’re hiring Software Engineers, Product Managers, Researchers, and more!

https://octoml.ai/careers

Or email me: areusch@octoml.ai
Thank you!

- Much of the work summarized today was authored and contributed by members of the TVM Community.

- microTVM would not be where it is today without their help
tinyML Summit 2022 Sponsors
Copyright Notice

This presentation in this publication was presented as a tinyML® Summit 2022. 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