μTVM
Deep Learning on Bare-Metal Devices
Pratyush Patel
No OS stack
Extend TVM to support bare-metal devices

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA  VTA

AutoTVM  AutoVTA

FPGA  ASIC

Hardware Fleet
Extend TVM to support bare-metal devices

Most bare-metal devices do not support LLVM
Extend TVM to support bare-metal devices

High-Level Differentiable IR
Tensor Expression IR

C, C++, LLVM, CUDA, VTA

Optimization

AutoTVM
AutoVTA
Hardware Fleet

FPGA, ASIC
Extend TVM to support bare-metal devices
Extend TVM to support bare-metal devices
μTVM builds upon AutoTVM

μTVM Runtime

- C Code Generator
- μDevice API

send program

optimize

run
A closer look at $\mu$TVM
A closer look at μTVM

μTVM Runtime

C Code Generator

μDevice API

telnet

OpenOCD

JTAG

run

IR->code

infer.c
A closer look at $\mu$TVM

C Code Generator

$\mu$Device API

OpenOCD

telnet

JTAG

IR $\rightarrow$ code

vendor

gcc

infer.c

infer.o

run
A closer look at μTVM

μTVM Runtime

C Code Generator

μDevice API

infer.c

infer.o

vendor gcc

1d linker

remap infer

OpenOCD

telnet

JTAG

run

IR->code
A closer look at μTVM

μTVM Runtime

- C Code Generator
- μDevice API

IR->code

Vendor gcc

infer.c -> infer.o

1d linker

remap infer

OpenOCD

telnet

custom loader

JTAG

run
A closer look at μTVM

μTVM Runtime

C Code Generator

μDevice API

telnet

send

custom loader

OpenOCD

JTAG

program

run

IR->code

vendor gcc

infer.c

infer.o

ld linker

remap

infer
Next steps

- Iron out interfaces with actual hardware.
- Optimize with AutoTVM.
- Support restricted and configurable model sizes.
- Enable custom data types such as fixed-point precision formats.

Get in touch!
Pratyush Patel – patelp1@cs.uw.edu