TVM Stack Overview

Tianqi Chen
Beginning of TVM Story
Beginning of TVM Story
Beginning of TVM Story
Beginning of TVM Story

(a) Blocked convolution program with multiple thread contexts

(b) Convolution micro-coded program

(c) Max pool, batch norm and activation micro-coded program
Existing Approach

Frameworks

Hardware
Existing Approach

Frameworks

Hardware

High-level data flow graph
Existing Approach

Frameworks

High-level data flow graph

Primitive Tensor operators such as Conv2D

Hardware

NVIDIA
Existing Approach

High-level data flow graph

Primitive Tensor operators such as Conv2D

Offload to heavily optimized DNN operator library

eg. cuDNN

Frameworks

Hardware
Existing Approach: Engineer Optimized Tensor Operators
Existing Approach: Engineer Optimized Tensor Operators
Existing Approach: Engineer Optimized Tensor Operators

**Matmul: Operator Specification**

```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```
Existing Approach: Engineer Optimized Tensor Operators

Matmul: Operator Specification

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]
Existing Approach: Engineer Optimized Tensor Operators

Matmul: Operator Specification

\[
C = \text{tvm.compute}((m, n), \\
\text{lambda } y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

Vanilla Code

```python
for y in range(1024):
    for x in range(1024):
        C[y][x] = 0
    for k in range(1024):
        C[y][x] += A[k][y] \times B[k][x]
```
Existing Approach: Engineer Optimized Tensor Operators

Matmul: Operator Specification

\[
C = tvm.compute((m, n),
\text{lambda } y, x: tvm.sum(A[k, y] \times B[k, x], axis=k))
\]

Loop Tiling for Locality

```python
for yo in range(128):
    for xo in range(128):
        C[yo*8:yo*8+8][xo*8:xo*8+8] = 0
    for ko in range(128):
        for yi in range(8):
            for xi in range(8):
                for ki in range(8):
                    C[yo*8+yi][xo*8+xi] +=
                        A[ko*8+ki][yo*8+yi] \times B[ko*8+ki][xo*8+xi]
```
Existing Approach: Engineer Optimized Tensor Operators

**Matmul: Operator Specification**

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k))
\]

**Map to Accelerators**

```python
inp_buffer AL[8][8], BL[8][8]
acc_buffer CL[8][8]
for yo in range(128):
    for xo in range(128):
        vdla.fill_zero(CL)
        for ko in range(128):
            vdla.dma_copy2d(AL, A[ko*8:ko*8+8][yo*8:yo*8+8])
            vdla.dma_copy2d(BL, B[ko*8:ko*8+8][xo*8:xo*8+8])
            vdla.fused_gemm8x8_add(CL, AL, BL)
        vdla.dma_copy2d(C[yo*8:yo*8+8,xo*8:xo*8+8], CL)
```

Human exploration of optimized code
Limitations of Existing Approach

Frameworks

cuDNN

NVIDIA
Limitations of Existing Approach

Frameworks

cuDNN

NVIDIA
Limitations of Existing Approach

Frameworks

cuDNN

NVIDIA
Limitations of Existing Approach

cuDNN

Frameworks

[Images of various frameworks and NVIDIA GPU]
Limitations of Existing Approach

cuDNN

Frameworks: TensorFlow, ML, K, m

NVIDIA
Limitations of Existing Approach

Frameworks

New operator introduced by operator fusion optimization potentially benefit: 1.5x speedup

cuDNN

NVIDIA
Limitations of Existing Approach

New operator introduced by operator fusion optimization potentially benefit: 1.5x speedup

cuDNN

Frameworks

NVIDIA
Limitations of Existing Approach

New operator introduced by operator fusion optimization potentially benefit: 1.5x speedup
Limitations of Existing Approach

New operator introduced by operator fusion optimization potentially benefit: 1.5x speedup

Engineering intensive
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware
Learning-based Learning System

Hardware aware Search Space of Optimized Tensor Programs
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

directly generate optimized program for new operator workloads and hardware

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware
Hardware-aware Search Space

Tensor Expression Language (Specification)

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]
Hardware-aware Search Space

Tensor Expression Language (Specification)

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

Define search space of hardware aware mappings from expression to hardware program

Based on Halide’s compute/schedule separation

Hardware
Hardware-aware Search Space

CPUs

Compute Primitives
- scalar
- vector

Memory Subsystem
- L3
- L2
- L1D
- L1I
- implicitly managed

Loop Transformations
Cache Locality
Vectorization

Reuse primitives from prior work:
Halide, Loopy
Challenge to Support Diverse Hardware Backends

- CPUs
- GPUs
- TPU-like specialized Accelerators
Hardware-aware Search Space

GPUs

Compute Primitives
- scalar
- vector

Memory Subsystem
- L2
- SM
- TX/L1
- RF
- mixed
Hardware-aware Search Space

GPUs

Compute Primitives

- scalar
- vector

Memory Subsystem

- L2
- SM
- TX/L1
- RF

mixed

Shared memory among compute cores
Hardware-aware Search Space

**GPUs**

**Compute Primitives**
- scalar
- vector

**Memory Subsystem**
- L2
- SM
- TX/L1
- RF

**Shared memory among compute cores**

**Use of Shared Memory**

**Thread Cooperation**
Hardware-aware Search Space

TPU-like Specialized Accelerators

Compute Primitives

Memory Subsystem

Unified Buffer

Explicitly managed
Hardware-aware Search Space

TPU-like Specialized Accelerators

Compute Primitives

tensor

Memory Subsystem

Unified Buffer

FIFO

Acc

explicitly managed
Tensorization Challenge

Compute primitives
Tensorization Challenge

Compute primitives

scalar
Tensorization Challenge

Compute primitives

scalar

vector
Tensorization Challenge

Compute primitives

- scalar
- vector
- tensor
Tensorization Challenge

Hardware designer: declare tensor instruction interface with Tensor Expression

```python
w, x = t.placeholder((8, 8)), t.placeholder((8, 8))
k = t.reduce_axis((0, 8))
y = t.compute((8, 8), lambda i, j: t.sum(w[i, k] * x[j, k], axis=k))

def gemm_intrin_lower(inputs, outputs):
    ww_ptr = inputs[0].access_ptr("r")
    xx_ptr = inputs[1].access_ptr("r")
    zz_ptr = outputs[0].access_ptr("w")
    compute = t.hardware_intrin("gemm8x8", ww_ptr, xx_ptr, zz_ptr)
    reset = t.hardware_intrin("fill_zero", zz_ptr)
    update = t.hardware_intrin("fuse_gemm8x8_add", ww_ptr, xx_ptr, zz_ptr)
    return compute, reset, update

gemm8x8 = t.decl_tensor_intrin(y.op, gemm_intrin_lower)
```
Tensorization Challenge

Hardware designer:
declare tensor instruction interface
with Tensor Expression

\[
w, x = \text{t.placeholder}((8, 8)), \text{t.placeholder}((8, 8))
\]

\[
k = \text{t.reduce_axis}((0, 8))
\]

\[
y = \text{t.compute}((8, 8), \lambda i, j: \text{t.sum}(w[i, k] \times x[j, k], \text{axis}=k))
\]

\[
def \text{gemm_intrin_lower}(\text{inputs}, \text{outputs}):
  \text{ww_ptr} = \text{inputs}[0].\text{access_ptr}("r")
  \text{xx_ptr} = \text{inputs}[1].\text{access_ptr}("r")
  \text{zz_ptr} = \text{outputs}[0].\text{access_ptr}("w")
  \text{compute} = \text{t.hardware_intrin}("gemm8x8", \text{ww_ptr}, \text{xx_ptr}, \text{zz_ptr})
  \text{reset} = \text{t.hardware_intrin}("fill_zero", \text{zz_ptr})
  \text{update} = \text{t.hardware_intrin}("fuse_gemm8x8_add", \text{ww_ptr}, \text{xx_ptr}, \text{zz_ptr})
  \text{return compute, reset, update}
\]

gemm8x8 = \text{t.decl_tensor_intrin}(y.\text{op}, \text{gemm_intrin_lower})

Tensorize:
transform program
to use tensor instructions
Hardware-aware Search Space

TPU-like Specialized Accelerators

Compute Primitives

tensor

Memory Subsystem

Unified Buffer
FIFO
Acc
explicitly managed
Hardware-aware Search Space

TPU-like Specialized Accelerators

Compute Primitives

Memory Subsystem

Unified Buffer

FIFO

Acc

explicitly managed

tensor
Hardware-aware Search Space

Tensor Expression Language

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k)) \]
Hardware-aware Search Space

Tensor Expression Language

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \times B[k, x], \text{axis}=k))
\]

Primitives in prior work: Halide, Loopy

Loop Transformations
Thread Bindings
Cache Locality

Hardware
Hardware-aware Search Space

Tensor Expression Language

\[
C = \text{tvm.compute}(\{m, n\}, \\
\quad \text{lambda } y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k))
\]

Primitives in prior work:
Halide, Loopy

New primitives for GPUs,
and enable TPU-like Accelerators

Loop Transformations
Thread Bindings
Cache Locality
Thread Cooperation
Tensorization
Latency Hiding

Hardware
Hardware-aware Search Space

Tensor Expression Language

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k))
\]
Hardware-aware Search Space

Tensor Expression Language

\[ C = \text{tvm.compute}((m, n), \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis}=k)) \]

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding

Hardware:
Hardware-aware Search Space

Tensor Expression Language

\[
C = \text{tvm.compute}((m, n), \\
\quad \lambda y, x: \text{tvm.sum}(A[k, y] \ast B[k, x], \text{axis} = k))
\]

Billions of possible optimization choices

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

Hardware
Learning-based Learning System

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

AutoTVM: Machine Learning based Program Optimizer

Hardware
Some Quick Results
End to End Inference Performance (Nvidia Titan X)

- Tensorflow
- Apache MXNet
- Tensorflow-XLA

![Graph showing performance comparison between different models and frameworks.](image-url)
End to End Inference Performance (Nvidia Titan X)

Backed by cuDNN

- Orange: Tensorflow
- Green: Apache MXNet
- Red: Tensorflow-XLA

*Time (ms)*

- ResNet-18
- MobileNet
- LSTM LM
- DQN
- DCGAN
End to End Inference Performance (Nvidia Titan X)

- Tensorflow
- Apache MXNet
- TVM: without graph optimizations
- Tensorflow-XLA

![Graph showing performance comparison](image-url)
End to End Inference Performance (Nvidia Titan X)

- **Tensorflow**
- **Apache MXNet**
- **TVM: without graph optimizations**
- **TVM: all optimizations**

![Graph showing inference performance for different models and frameworks.](image-url)
End to End Inference Performance (Nvidia Titan X)

- Tensorflow
- Apache MXNet
- Tensorflow-XLA
- TVM: without graph optimizations
- TVM: all optimizations

Competitive on standard models
End to End Inference Performance (Nvidia Titan X)

- Tensorflow
- Apache MXNet
- TVM: without graph optimizations
- TVM: all optimizations

ResNet-18
MobileNet

LSTM LM
DQN
DCGAN

Competitive on standard models
Bigger gap on less conventional models
Across Hardware Platforms

ARM CPU(A53) vs ARM GPU(MALI)
Supporting New Specialized Accelerators

Frameworks

High-level data flow graph and optimizations

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

LLVM

CUDA
Supporting New Specialized Accelerators

- Frameworks
- High-level data flow graph and optimizations
- Hardware aware Search Space of Optimized Tensor Programs
- Machine Learning based Program Optimizer
- LLVM
- CUDA
- VTA: Open, Customizable Deep Learning Accelerator
- Edge FPGA
- Data Center FPGA
- ASIC
More on the High-Level Optimizations

- Frameworks
- High-level data flow graph and optimizations
- Hardware aware Search Space of Optimized Tensor Programs
- Machine Learning based Program Optimizer
- LLVM
- CUDA
- VTA: Open, Customizable Deep Learning Accelerator
  - Edge FPGA
  - Data Center FPGA
  - ASIC
- ASIC
More on the High-Level Optimizations

Frameworks

Hardware aware Search Space of Optimized Tensor Programs

Machine Learning based Program Optimizer

LLVM  CUDA  VTA: Open, Customizable
Deep Learning Accelerator

Edge FPGA  Data Center FPGA  ASIC
More on the High-Level Optimizations

- **Frameworks**
  - TensorFlow
  - ML
  - Relay: High-Level Differentiable IR
  - Machine Learning based Program Optimizer

- **Hardware**
  - LLVM
  - CUDA
  - VTA: Open, Customizable Deep Learning Accelerator
  - Edge FPGA
  - Data Center FPGA
  - ASIC

- **Platforms**
  - Deep Learning Accelerator
  - Data Center FPGA
  - Edge FPGA
  - ASIC
TVM: Learning-based Deep Learning Compiler
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal

VTA

Edge FPGA
Cloud FPGA
ASIC
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal

VTA

AutoTVM

Edge FPGA

Cloud FPGA

ASIC

Optimization

Hardware Fleet
TVM: Learning-based Deep Learning Compiler

- High-Level Differentiable IR
- Tensor Expression IR
- LLVM, CUDA, Metal
- VTA
- AutoTVM
- AutoVTA
- Hardware Fleet
- Optimization
- Edge FPGA
- Cloud FPGA
- ASIC
TVM: Learning-based Deep Learning Compiler
TVM Open Source Community

- Prefer public archivable discussion
- Open RFC discussion
- Bring in new members by merit

https://docs.tvm.ai/contribute/community.html
TVM: Learning-based Deep Learning Compiler
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal  VTA

Edge FPGA  Cloud FPGA  ASIC
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal

VTA

Optimization

AutoTVM

AutoVTA

Hardware Fleet

Edge FPGA

Cloud FPGA

ASIC
TVM: Learning-based Deep Learning Compiler

High-Level Differentiable IR

Tensor Expression IR

LLVM, CUDA, Metal
VTA

Optimization

AutoTVM
AutoVTA

Hardware Fleet

Edge FPGA
Cloud FPGA
ASIC
TVM: Learning-based Deep Learning Compiler

- High-Level Differentiable IR
- Tensor Expression IR
- LLVM, CUDA, Metal
- VTA
- Edge FPGA, Cloud FPGA, ASIC
- AutoTVM
- AutoVTA
- Hardware Fleet