TensorIR: An Abstraction for Automatic Tensorized Program Optimization

Siyuan Feng*
Shanghai Jiao Tong University
Shanghai, China
hzfengsy@sjtu.edu.cn

Bohan Hou*
Carnegie Mellon University
Pittsburgh, USA
bohanhou@cs.cmu.edu

Hongyi Jin†
Carnegie Mellon University
Pittsburgh, USA
hongyij@cs.cmu.edu

Wuwei Lin
OctoML
Seattle, USA
wlin@octoml.ai

Junru Shao
OctoML
Seattle, USA
jshao@octoml.ai

Ruihang Lai†
Carnegie Mellon University
Pittsburgh, USA
ruihangl@cs.cmu.edu

Zihao Ye
University of Washington
Seattle, USA
zhye@cs.washington.edu

Lianmin Zheng
UC Berkeley
Berkeley, USA
lmzheng@berkeley.edu

Cody Hao Yu
Amazon Web Services
Seattle, USA
hyuz@amazon.com

Yong Yu
Shanghai Jiao Tong University
Shanghai, China
yyu@apex.sjtu.edu.cn

Tianqi Chen
Carnegie Mellon University, OctoML
Pittsburgh, USA
tqchen@cmu.edu
tqchen@octoml.ai

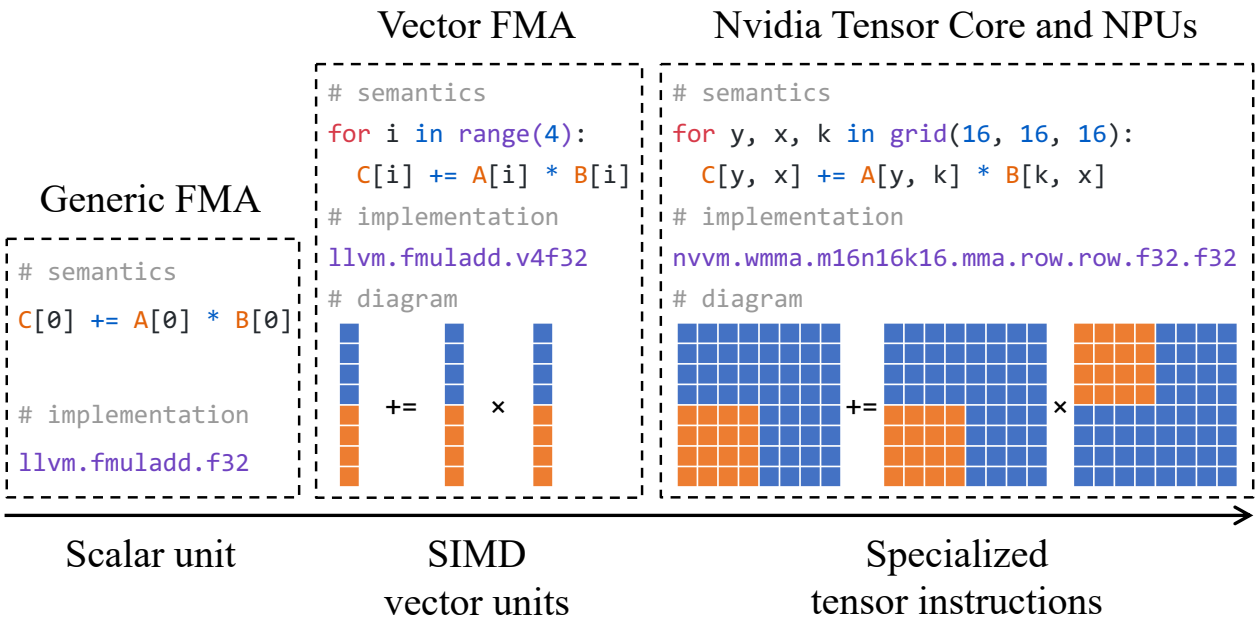
中山大學
SUN YAT-SEN UNIVERSITY

主要内容

中山大學
SUN YAT-SEN UNIVERSITY

- There are increasing demands to deploy smart applications to a broad spectrum of devices ranging from servers to embedded environments.
- Deploying high-performance machine learning models has become an emerging challenge in various areas.

- Trends of hardware specialization.
- Most of tensorized program optimization are optimized by domain experts.
- We need a automatic compilation approach.



Challenge

- Abstraction for Tensorized Programs
  - Previous tensor compilers rely on polyhedral Compilation, schedule tree data structure and the corresponding lowering rule.
- Large Design Space of Possible Tensorized Program Optimizations
  - The combinations of code transformations form a large search space.
  - These transformations need to be made in conjunction with tensorized computations, bringing additional complexities to analysis and automation.

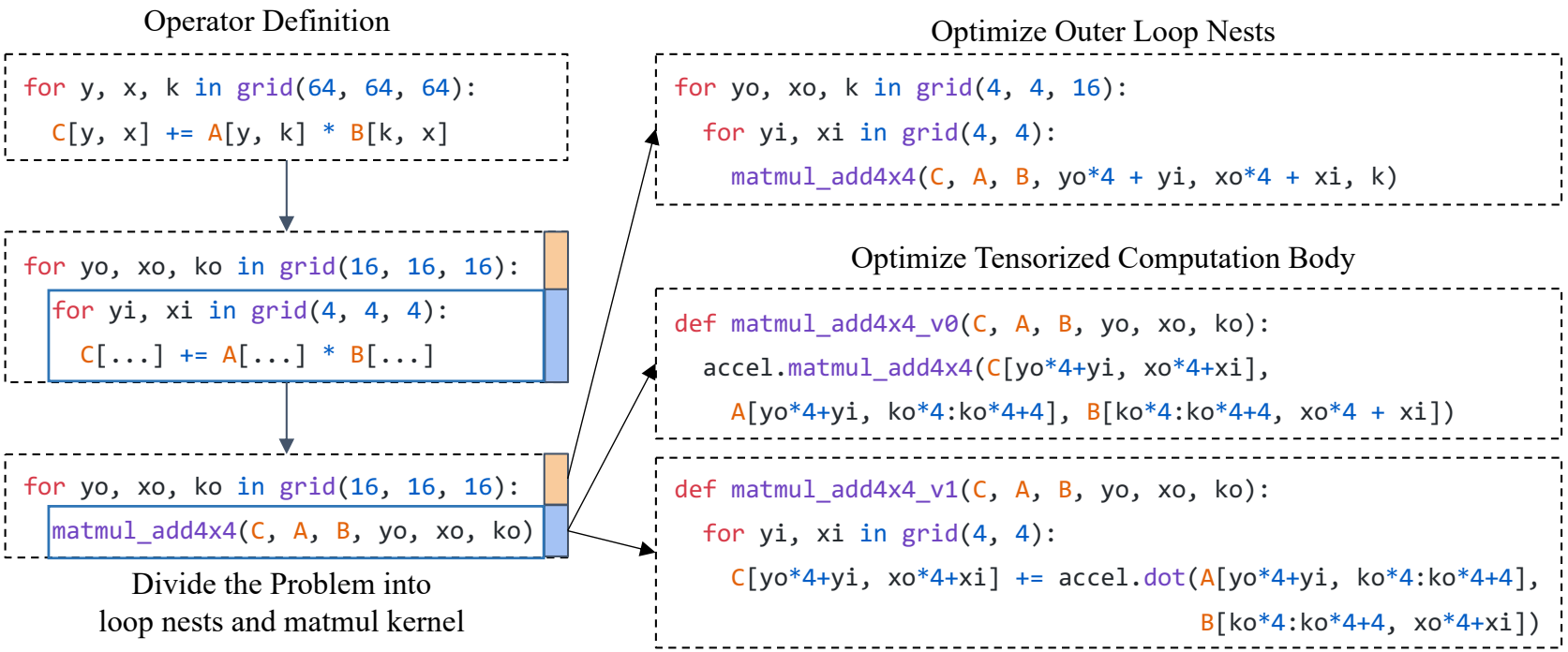
Contributions

- TE and loop nests $\rightarrow$ TensorIR
  - Block: Divide and isolate tensorized computation region from the outer loop nests.
- schedule $\rightarrow$ transformation primitives
- tensorization-aware automatic scheduler

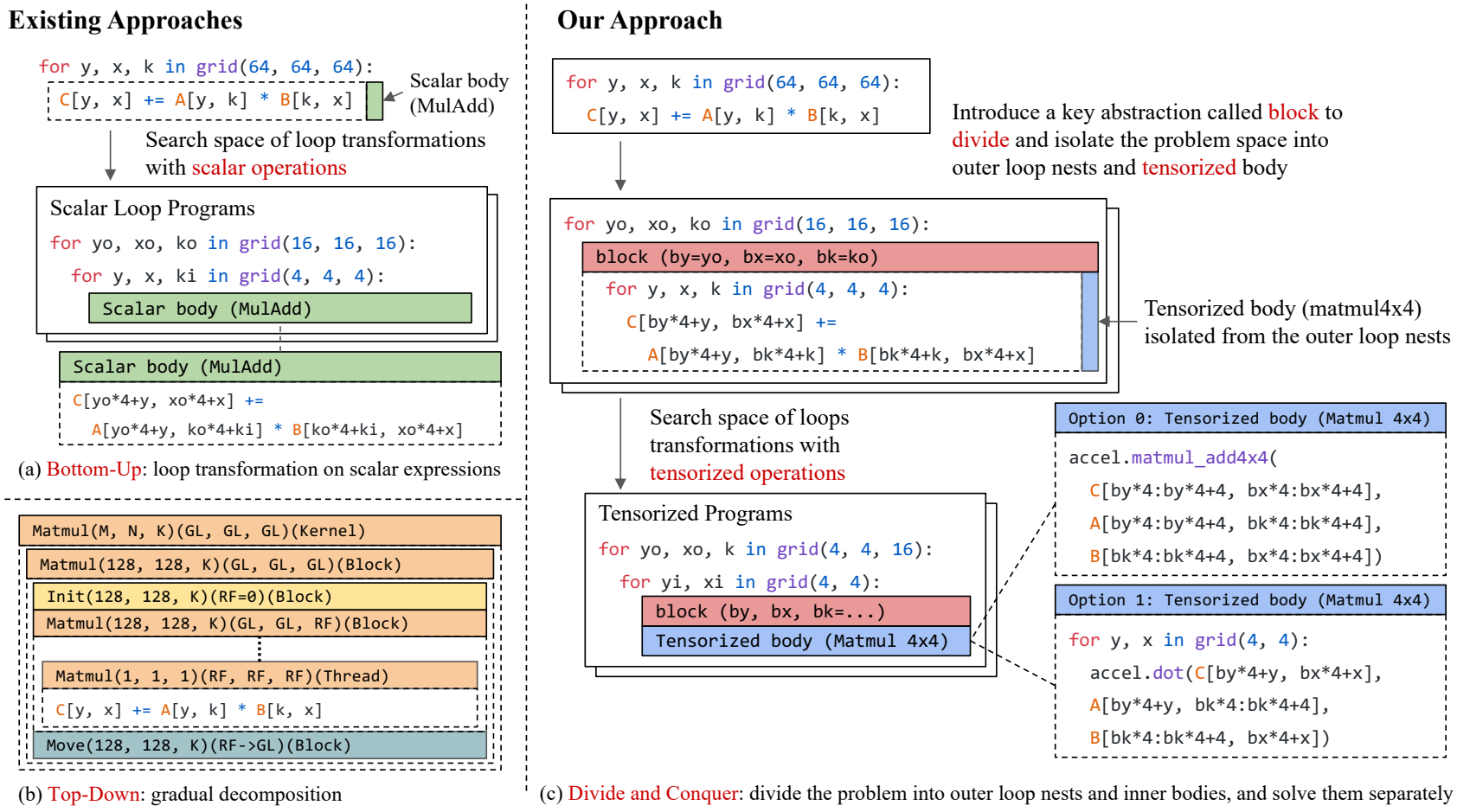
主要内容

Motivation

- Expert developer



Overview


Existing Approaches
for y, x, k in grid(64, 64, 64):
C[y, x] += A[y, k] * B[k, x]
Scalar body (MulAdd)
Search space of loop transformations with scalar operations
Scalar Loop Programs
for yo, xo, ko in grid(16, 16, 16):
for y, x, ki in grid(4, 4, 4):
Scalar body (MulAdd)
Scalar body (MulAdd)
C[yo*4+y, xo*4+x] +=
A[yo*4+y, ko*4+ki] * B[ko*4+ki, xo*4+x]
(a) Bottom-Up: loop transformation on scalar expressions
Matmul(M, N, K)(GL, GL, GL)(Kernel)
Matmul(128, 128, K)(GL, GL, GL)(Block)
Init(128, 128, K)(RF=0)(Block)
Matmul(128, 128, K)(GL, GL, RF)(Block)
Matmul(1, 1, 1)(RF, RF, RF)(Thread)
C[y, x] += A[y, k] * B[k, x]
Move(128, 128, K)(RF->GL)(Block)
(b) Top-Down: gradual decomposition
Our Approach
for y, x, k in grid(64, 64, 64):
C[y, x] += A[y, k] * B[k, x]
Introduce a key abstraction called block to divide and isolate the problem space into outer loop nests and tensorized body
for yo, xo, ko in grid(16, 16, 16):
block (by=yo, bx=xo, bk=ko)
for y, x, k in grid(4, 4, 4):
C[by*4+y, bx*4+x] +=
A[by*4+y, bk*4+k] * B[bk*4+k, bx*4+x]
Tensorized body (matmul4x4) isolated from the outer loop nests
Search space of loops transformations with tensorized operations
Tensorized Programs
for yo, xo, k in grid(4, 4, 16):
for yi, xi in grid(4, 4):
block (by, bx, bk=...)
Tensorized body (Matmul 4x4)
Option 0: Tensorized body (Matmul 4x4)
accel.matmul_add4x4(
C[by*4:by*4+4, bx*4:bx*4+4],
A[by*4:by*4+4, bk*4:bk*4+4],
B[bk*4:bk*4+4, bx*4:bx*4+4])
Option 1: Tensorized body (Matmul 4x4)
for y, x in grid(4, 4):
accel.dot(C[by*4+y, bx*4+x],
A[by*4+y, bk*4:bk*4+4],
B[bk*4:bk*4+4, bx*4+x])
(c) Divide and Conquer: divide the problem into outer loop nests and inner bodies, and solve them separately

TensorIR and Block Structure

Computation: `C = exp(A + 1)`

```
@script
def fuse_add_exp(
    A: Buffer[(64, 64), "float32"],
    C: Buffer[(64, 64), "float32"],
):
    B = alloc_buffer((64, 64), "float32")
    for i, j in grid(64, 64):
        with block("block_B"):
            vi = spatial_axis(64, i)
            vj = spatial_axis(64, j)
            B[vi, vj] = A[vi, vj] + 1
    for i in range(64):
        with block("block_C"):
            vi = spatial_axis(64, i)
            for j in range(64):
                C[vi, j] = exp(B[vi, j])
```

Multi-dimensional **buffer**

Loop nests

Computational **block**

```
for yo, xo, ko in grid(16, 16, 16):
  with block():
    vy, vx, vk = ...
    block_siguatures
    with init():
      for y, x in grid(4, 4):
        C[vy*4+y, vx*4+x] = 0.0
    for y, x, k in grid(4, 4, 4):
      C[vy*4+y, vx*4+x] +=
        A[vy*4+y, vk*4+k] * B[vk*4+k, vx*4+x]
```

outer loop

signatures

init stmt

body

Block Signature

Iterator domain and binding values

```
vy: spatial_axis(length=16, binding_value=yo)
vx: spatial_axis(length=16, binding_value=xo)
vk: reduce_axis (length=16, binding_value=ko)
```

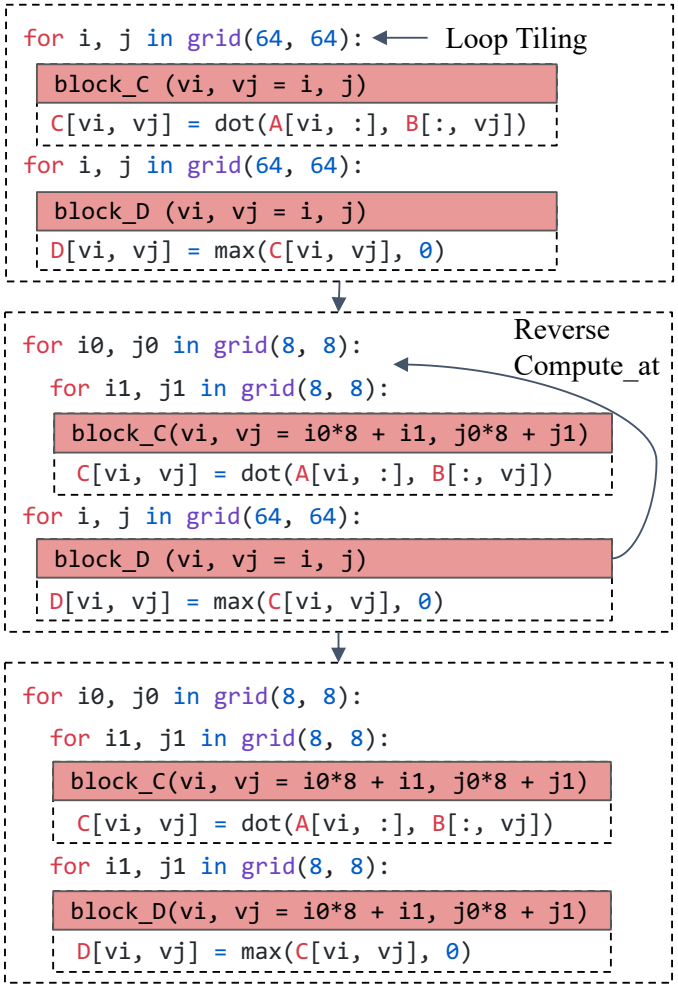
Producer-consumer dependency relations

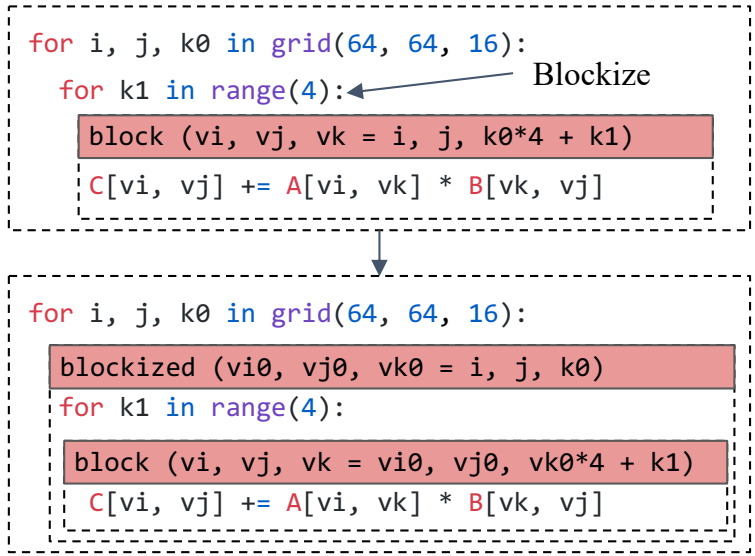
```
read  A[vy*4:vy*4 + 4, vk*4:vk*4 + 4]
read  B[vk*4:vk*4 + 4, vx*4:vx*4 + 4]
write C[vy*4:vy*4 + 4, vx*4:vx*4 + 4]
```

Schedule

- schedule
- separation of scheduling and TensorIR
- schedulable

Loop Transformations and Blockization

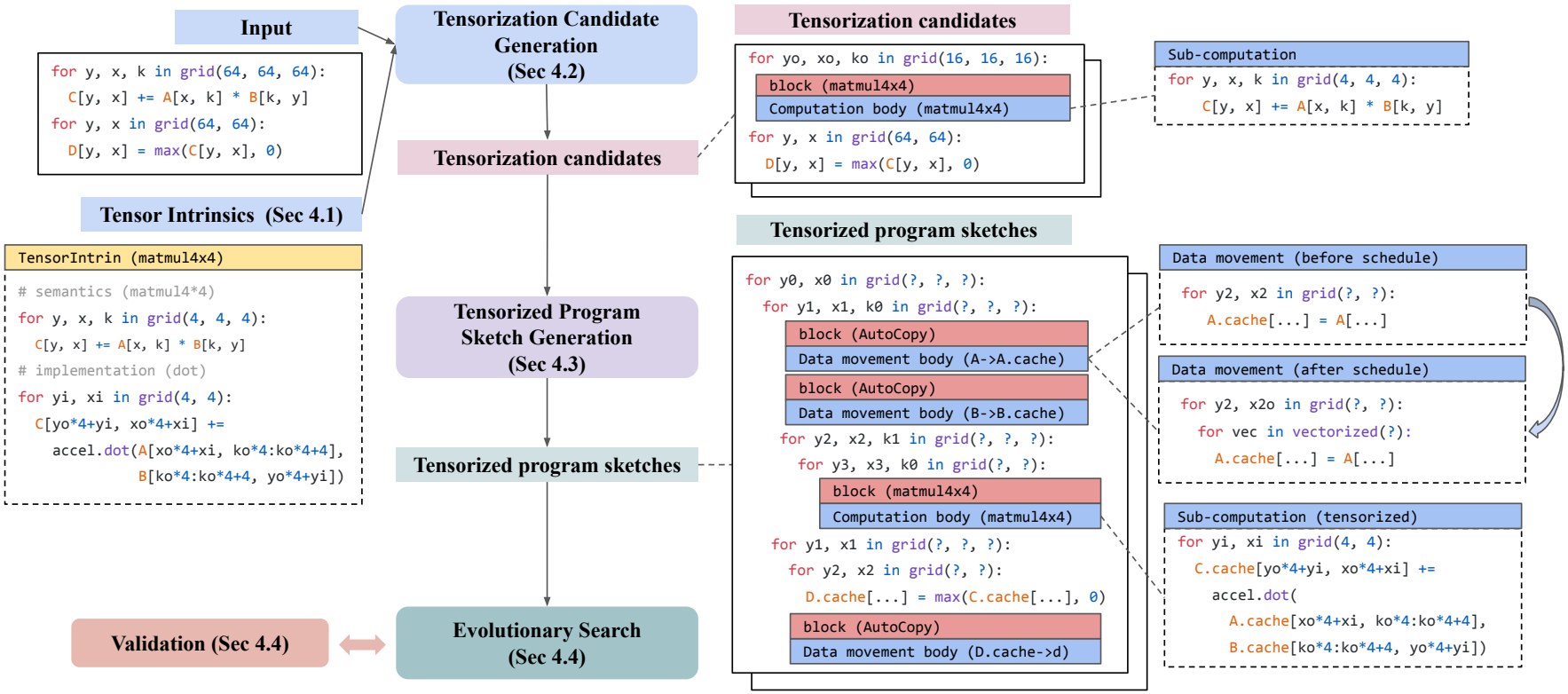




主要内容

中山大學
SUN YAT-SEN UNIVERSITY

Automatic optimization Overview



Tensorization Candidate Generation

Conv2D

```
for n, h, w, c, rh, rw, rc in grid(...):
  C[n, h, w, c] +=
    A[n, h*2+rh, w*2+rw, rc]
  * B[c, rh, rw, rc]
```

TensorIntrin

```
for x, y, k in grid(16, 16, 16):
  C[x, y] += A[x, k] * B[y, k]
```

Conv2D (ReIndexed)

```
# ReIndex Block for A
for n, h, w, c, rh, rw, rc in grid(...):
  Ar[n, h, w, rh, rw, rc]
    = A[n, h*2+rh, w*2+rw, rc]
# ReIndex Block for B
for c, rh, rw, rc in grid(...):
  Br[c, rh, rw, rc] = B[c, rh, rw, rc]
# ReIndexed Conv2D
for n, h, w, c, rh, rw, rc in grid(...):
  Cr[n, h, w, c] += Ar[n, h, w, rh, rw, rc]
                  * Br[c, rh, rw, rc]
# ReIndex Block for C
for n, h, w, c in grid(...):
  C[n, h, w, c] = Cr[n, h, w, c]
```

Conv2D (ReIndexed)

```
Cr[n, h, w, c] +=
  Ar[n, h, w, rh, rw, rc]
* Br[c, rh, rw, rc]
```

$\chi(n) = \chi(h) = \chi(w) = [1,1,0]$
$\chi(c) = [1,0,1]$
$\chi(rh) = \chi(rw) = \chi(rc) = [0,1,1]$

TensorIntrin

```
C[x, y] += A[x, k] * B[y, k]
```

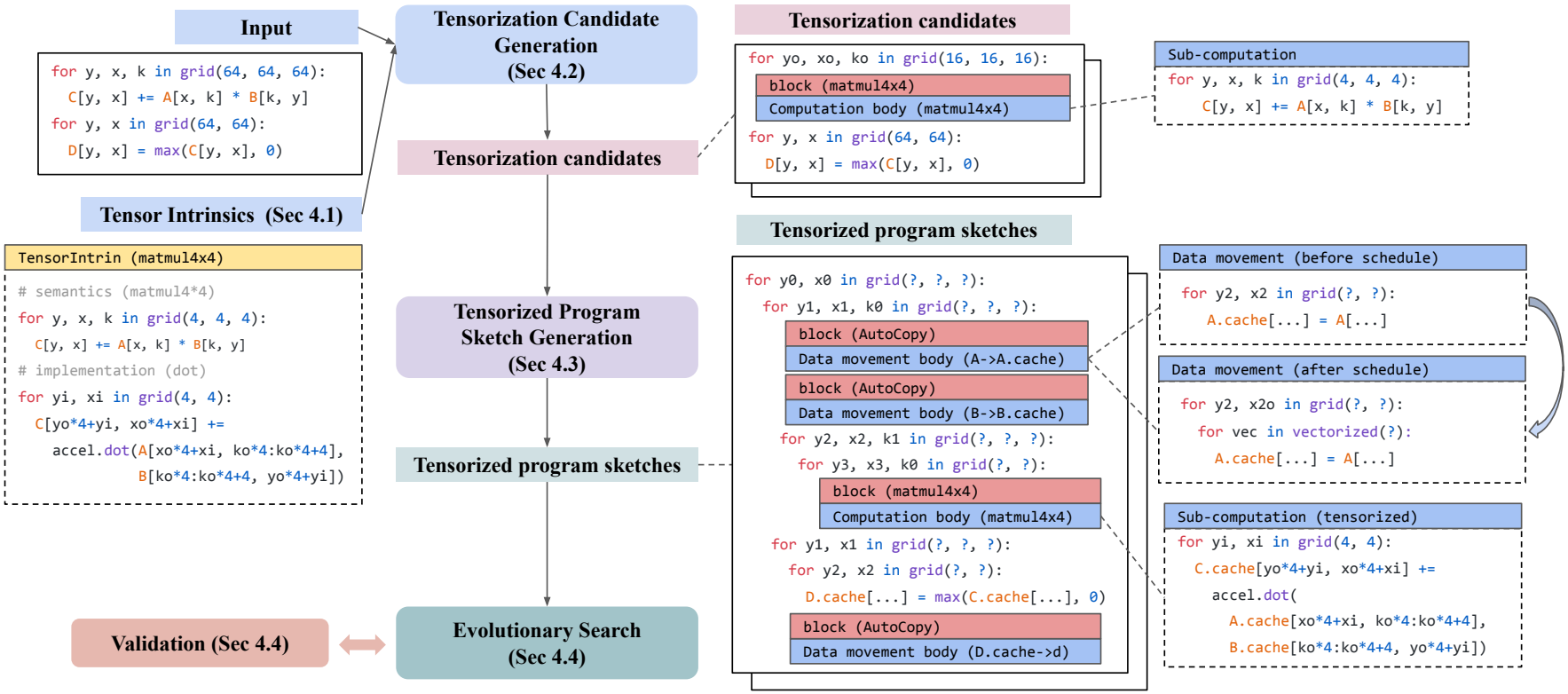
$\chi(x) = [1,1,0]$
$\chi(y) = [1,0,1]$
$\chi(k) = [0,1,1]$

Conv2D (Layout Transformed)

```
# Layout Transformation Block for A
for n, h, w, c, rh, rw, rc in grid(...):
  At[n*h*w, rh*rw*rc]
    = A[n, h*2+rh, w*2+rw, rc]
# Layout Transformation Block for B
for c, rh, rw, rc in grid(...):
  Bt[c, rh*rw*rc] = B[c, rh, rw, rc]
# Transformed Conv2D
for nhw, c, rhwc in grid(...):
  Ct[nhw, c] += At[nhw, rhwc] * Bt[c, rhwc]
# Layout Transformation Block for C
for n, h, w, c in grid(...):
  C[n, h, w, c] = Ct[n*h*w, c]
```

ReIndex → Propose Match → TransformLayout

Tensorized Program Sketch Generation

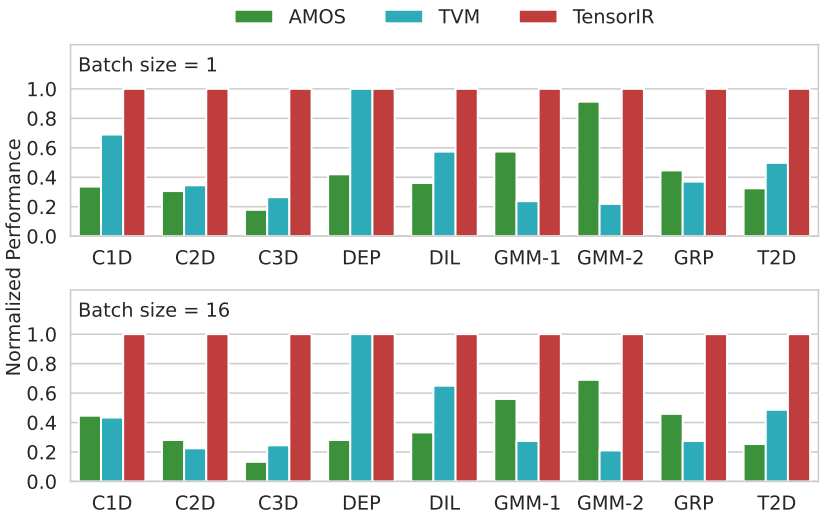


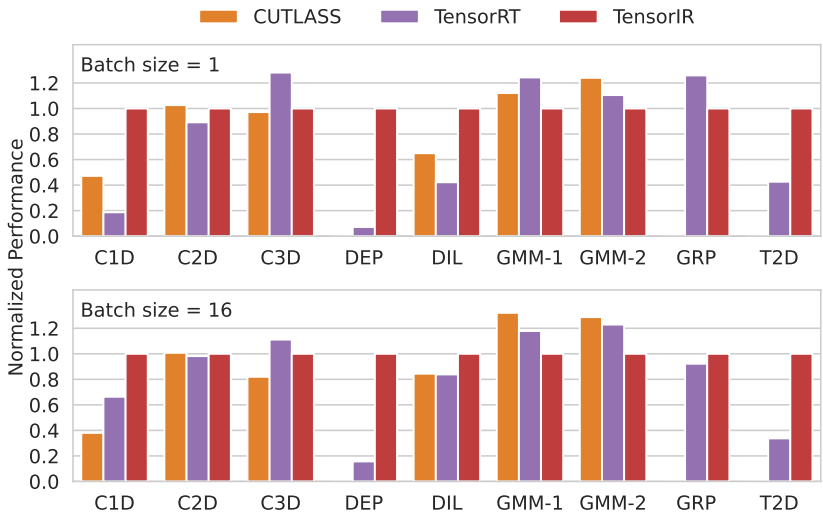
主要内容

中山大學
SUN YAT-SEN UNIVERSITY

Single Operator Evaluation

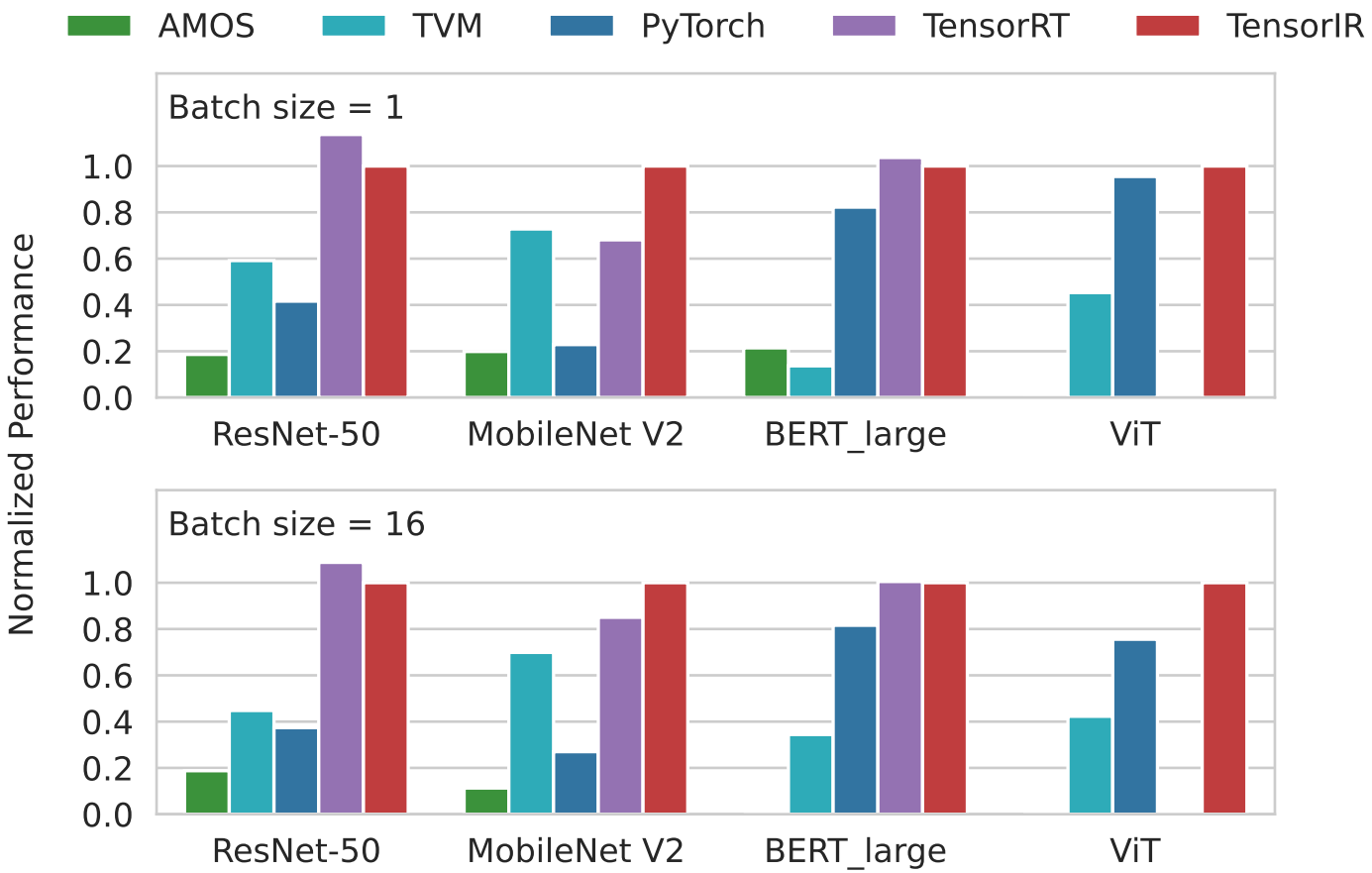
- RTX3080
- depthwise convolution (DEP), dilated convolution (DIL), group convolution (GRP), and transposed 2D convolution (T2D)





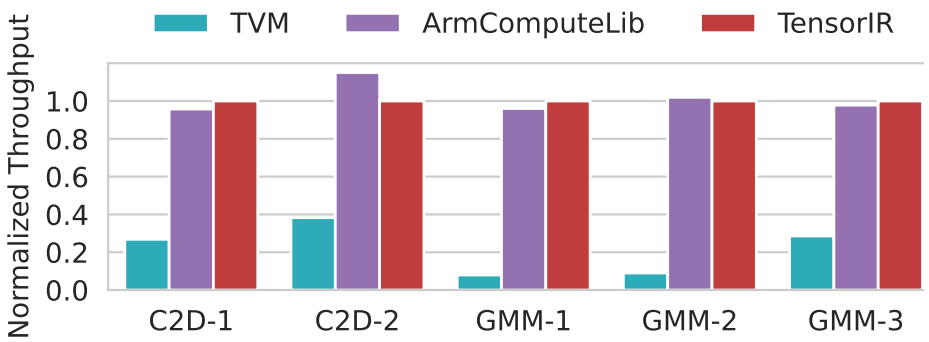
End-to-end Model Evaluation

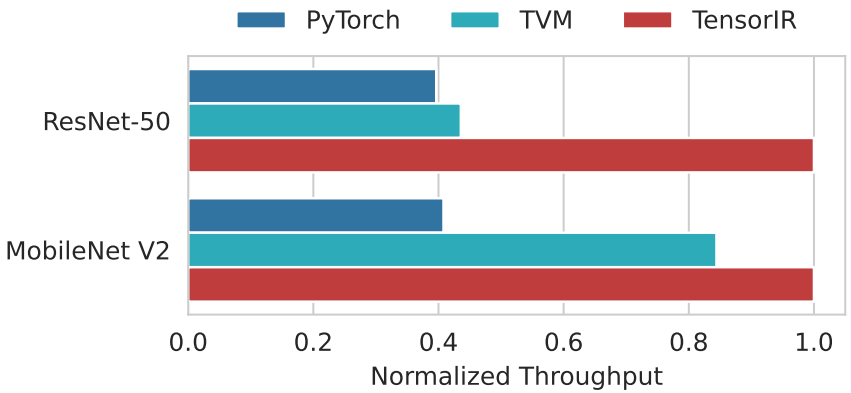
- RTX3080



ARM CPU Evaluation

- AWS Graviton2 CPU





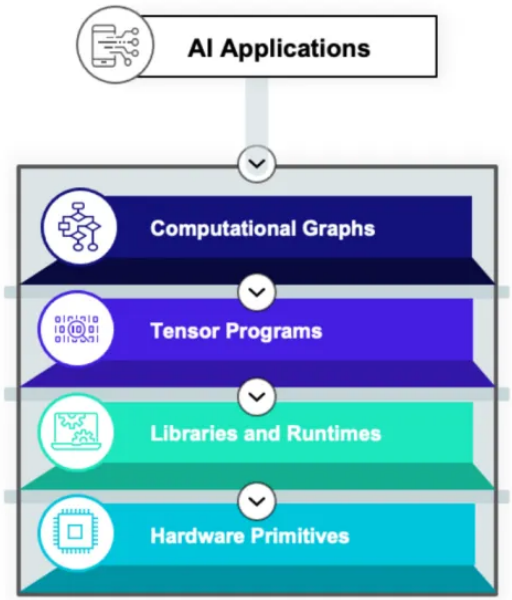
主要内容

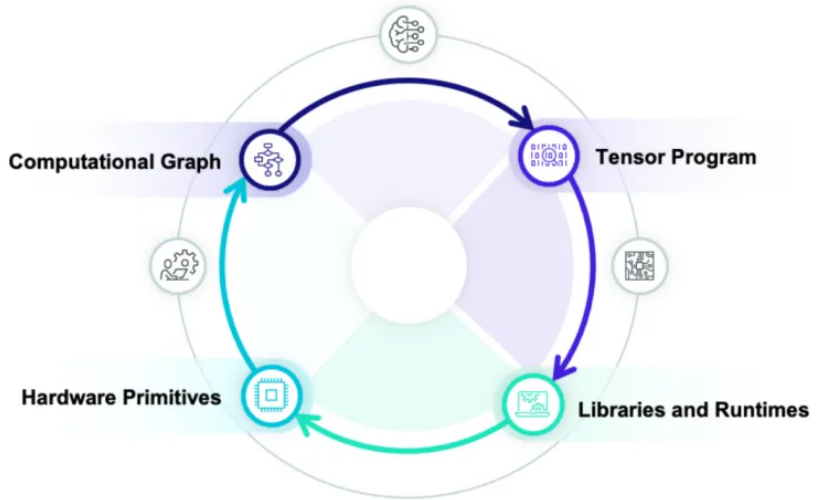
Conclusion

- TensorIR, an abstraction for automatic tensorized program optimization

TVM

- TVM Unity
- TE→TensorIR
- Relay→Relax





Reference

- paper: https://dl.acm.org/doi/abs/10.1145/3575693.3576933
- MLC: https://mlc.ai/zh/chapter_tensor_program/case_study.html
- TVM Discuss: https://discuss.tvm.apache.org/t/rfc-tensorir-a-schedulable-ir-for-tvm/7872
- TVM TensorIR Talk: https://www.youtube.com/watch?v=yaf2aAAz2oQ
- TVM Unity: https://zhuanlan.zhihu.com/p/446935289
- Relax: https://discuss.tvm.apache.org/t/relax-co-designing-high-level-abstraction-towards-tv 12496
- Polyhedral Compilation Overview: https://zhuanlan.zhihu.com/p/562552075