Tensor Program Optimization with Probabilistic Programs

Junru Shao, Xiyou Zhou, Siyuan Feng, Bohan Hou, Ruihang Lai, Hongyi Jin, Wuwei Lin, Masahiro Masuda, Cody Hao Yu, Tianqi Chen
Optimising Tensor Program

Key elements in automatic tensor program optimization

Initial Tensor Program $e_0$ → Search Space $S(e_0)$ → Search Algorithm → Optimal Program $e^*$

Examples

```
for i in range(1024):
    for j in range(1024):
        for k in range(1024):
            C[i, j] += A[i, k] * B[j, k]
```

```
for i0, j0 in grid(16, 8):
    for i1, j1 in grid(8, 16):
        for k0 in range(1024):
            for i2, j2 in grid(8, 8):
                C[... ] += ... 
```

```
for i0, j0 in grid(64, 8):
    for i1, j1 in grid(4, 32):
        for k0 in range(64):
            for i2, j2 in grid(4, 4):
                for k1 in range(16):
                    C[... ] += ...
```

…
Probabilistic Programming (PP)

- Probabilistic programming is **not** about writing software that behaves probabilistically.

- The key insight in PP is that **statistical modelling** can, when you do it enough, start to **feel a lot like programming**

- a probabilistic programming language is an ordinary programming language with `rand` and a great big pile of related tools that help you understand the program's statistical behaviour
Related work

• Halide (2013)
  • a language for fast, portable computation on images and tensors
• TVM (2018)
  • Template guided search space
• Ansor (2020)
  • Auto scheduler for subgraphs
• Difficult for expert to express domain knowledge in them
MetaSchedule, a domain-specific probabilistic programming language abstraction to construct a rich search space of tensor programs.
MetaSchedule, a domain-specific probabilistic programming language abstraction to construct a rich search space of tensor programs.
# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...
MetaSchedule Probabilistic language

# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...
MetaSchedule Probabilistic language

# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

Loop tiling
Loop fusion
MetaSchedule Probabilistic language

# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

Loop tiling

Loop fusion
MetaSchedule Probabilistic language

Tiling

# 1 Loop tiling for Dense

\(\theta_0, \theta_1 \sim \text{Sample-Tile}(i, \text{parts}=2)\)

\(\theta_2, \theta_3 \sim \text{Sample-Tile}(j, \text{parts}=2)\)

\(i_0, i_1 = \text{Split}(i, [\theta_0, \theta_1])\)

\(j_0, j_1 = \text{Split}(j, [\theta_2, \theta_3])\)

Reorder\((i_0, j_0, i_1, j_1)\)
# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

for i₀, j₀ in grid(θ₀, θ₂):
    for i₁, j₁ in grid(θ₁, θ₃):
        for k in range(16):
            C[...] += ...

Loop tiling
Loop fusion
# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

Loop tiling
for i0, j0 in grid(θ0, θ2):
    for i1, j1 in grid(θ1, θ3):
        for k in range(16):
            C[...] += ...

Loop fusion
MetaSchedule Probabilistic language

Fusion

# 2 ReLU fusion

$\theta_{\text{ReLU}} \sim \text{Sample–Compute–Location}(\text{ReLU})$

Compute–At($\text{ReLU}$, $\theta_{\text{ReLU}}$)
MetaSchedule Probabilistic language

# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

# Fused Dense + ReLU
# θReLU : Deep fusion under j1
for i0, j0 in grid(θ0, θ2):
    for i1, j1 in grid(θ1, θ3):
        for k in range(16):
            C[...] += ...
    for i', j' in grid(θ4, θ5):
        D[...] = ...

Loop tiling

Loop fusion

θReLU is j1
MetaSchedule Probabilistic language

# Dense:
for i in range(512):
    for j in range(256):
        for k in range(16):
            C[...] += ...

# ReLU:
for i' in range(512):
    for j' in range(256):
        D[...] = ...

# Fused Dense + ReLU
# ReLU : Shallow fusion under j0
for i0, j0 in grid(θ0, θ2):
    for i1, j1 in grid(θ1, θ3):
        for k in range(16):
            C[...] += ...
for i', j' in grid(θ4, θ5):
    D[...] = ...

Loop tiling

Loop fusion

θReLU is j0
Modularity

- Previous program transformation modularised
- Modules are composable
- Pre-written modules
  - By domain specialist
Learning driven approach

- End-to-end search
- Execution traces (validation)
Performance

Optimising End-to-End Deep Learning Models

![Graph showing normalized performance comparison between PyTorch, TVM, and MetaSchedule for different models on CPU and GPU.]
Performance

Composing modules

Performance with different search spaces.

BERT-Large Performance.
Thoughts & Critiques

Opinions are my own

• Why is probabilistic programming fit for search space construction
  • After all we can just do random sampling, which does not require PP
  • Guessing because it allows expressing of domain knowledge in a simple way
  • But this is more of PP’s contribution
• Evaluation comparing with hardware specific modules
  • Is this fair?
• Does not talk about searching time
Conclusions

- DSL (Probabilistic programming) for tensor program optimisation
- Rich search space construction
- Composable modules for program optimisation
- Learning driven approach based on search space specification


• Yan, Eddie Q. et al. “USING AUTOTVM TO AUTOMATICALLY GENERATE DEEP LEARNING LIBRARIES FOR MOBILE DEVICES.” (2019).

• Adrian Sampson, Probabilistic programming, url: http://adriansampson.net/doc/ppl.html, retrieved 26 Oct 2022