Ansor: Generating High-Performance Tensor Programs for Deep Learning

*R244: Large-Scale Data Processing and Optimisation*

Felix Jonathan Rocke

Improving the performance of deep learning models requires hardware-specific optimisations. Automatic code-generation (ML Compiler, e.g. AutoTVM) has the advantage of less to no engineering effort necessary to adapt to different/new hardware. However, the performance is not always as good as with manual optimisations. On the other hand, manual optimisation (Operator Library, e.g. CuDNN) requires significant engineering effort to adapt to different/new hardware, but the performance is often better than automatic generation.
Template Guided Search

1. Experts create hardware-specific tensor code templates
   Expert knowledge required in:
   • Hardware architecture
   • Optimisation techniques

2. Parameters are determined via an automatic search algorithm

```
for i.0 in range(?):
    for j.0 in range(?):
        for k.0 in range(?):
            for i.1 in range(?):
                for j.1 in range(?):
                    C[...] += A[...] * B[...]
        for i.2 in range(?):
            for j.2 in range(?):
                D[...] = max(C[...], 0.0)
```
Sequential Program Construction

- Programs are sequentially constructed through a fixed sequence of decisions.
- Uses unfolding rules for every node.
- Only the top-k candidates are kept.
- Cost function is used to evaluate incomplete programs.
  - Low accuracy of cost function at the beginning of program creation.
  - Candidate programs are pruned to early.
  - Limited search space.

**Beam Search with Early Pruning**

**Incomplete Program**

```python
for i in range(512):
    for j in range(512):
        D[...] = max(C[...], 0.0)
```

**How to build the next statement?**

- Candidate 1: Pruned
- Candidate 2: Kept
- Candidate 3: Kept
- Candidate 4: Pruned
1. High and low-level structures are separated
2. Create search space of high-level tensor programs
3. Sample high-level programs uniformly from search space
4. Sample low-level features
5. Fine-tune low-level features

- No early pruning or limited options
- Greater search space covered
- Better programs
Ansor Overview

1. Split computational graph into subgraphs
2. Schedule subgraph for program sampling
3. Uniformly select initial programs
4. Send programs to hardware for measurements

Provide feedback to the individual steps, e.g., updates to the learned cost model
GPU Benchmark Results

Results for Nvidia V100
Pros & Cons / Discussion

Cons:
- An ARMv8 CPU is used in the benchmarks
  - However, it is not specified if NEON is enabled, unlike for the x86 CPU, where it is stated that AVX-512 was used
- No multi-objective optimisations

Pros:
- Simplifies model optimisation since manual kernel or template development is replaced
- Better performance than hand-optimisation