Towards Automatic Function Call Generation for Deep Learning

Shizhi Tang (tsz19@mails.tsinghua.edu.cn) and Jidong Zhai (zhaijidong@tsinghua.edu.cn)
Tsinghua University

Implementing Deep Learning Models with Libraries is an Instruction Selection problem

A Deep Learning model = **description** + **computation**.

- **Description** → Directed Acyclic Graph (**DAG**)
- **Computation** → **Pre-optimized libraries** such as Intel MKL-DNN, Eigen, NVIDIA cuBLAS and/or NVIDIA cuDNN

Mapping a DAG to a function call sequence

=

A variant of Instruction Selection problem in traditional compiling

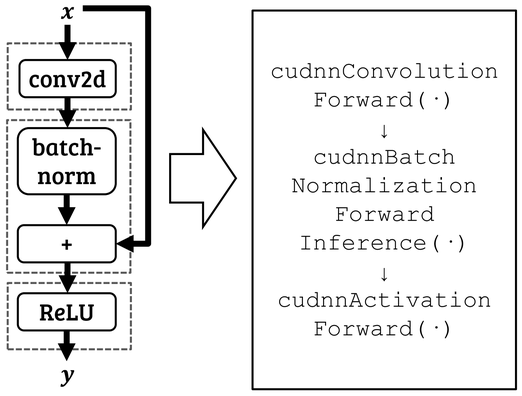

fig. 1. A mapping example.

1-node-to-N-function mapping is sub-optimal

- **Fused operations**
  - `cudnnConvolutionBiasActivationForward`
    `cudnnBatchNormalizationBackward`
- **In-place operations**
  - `cudnnActivationForward`
  - `cublas<t>`
- **Memory layouts**
  - NCHW / NHWC layout in CNN
  - Transpositions in `cublas<t>gemm`
  - Broadcasting in `cudnnAddTensor`
  - Blocked layouts such as "NC/32HW32"(cuDNN) or "nChw16c"(MKL-DNN)

An Exhaustive Search Algorithm

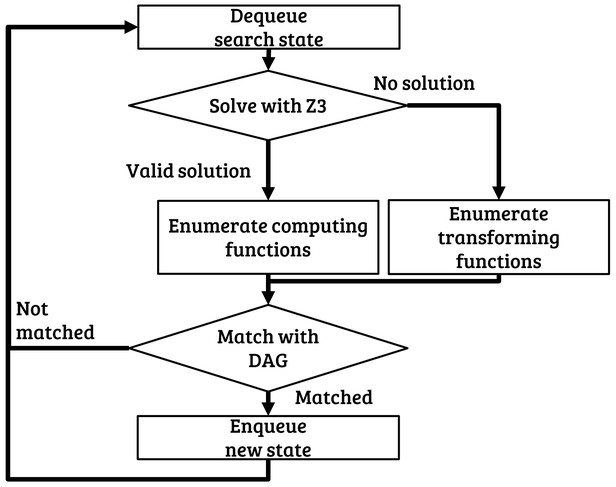

fig. 2. High-level search algorithm.

Checking the constraints between functions

We use **logic equations (solved by Z3)** to describe the constraints.

- Property of the tensors → Unknown predicate
- Constraint of each function in the libraries → First-order logic (in)equation set

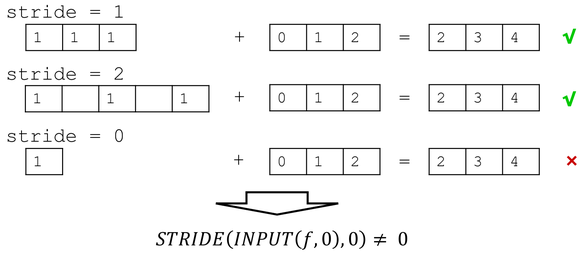

fig. 3. Level 1 functions in BLAS accepts vector with strides other than 1, i.e., two adjacent item in the vector may not be adjacent in the memory, BUT it requires strides not equal to 0.

Pruning the search

Dividing all functions into 2 categories

- **computing functions** actually perform operations.
- **transforming functions** only transform the tensor from one layout to another.

Assume we only need transforming functions when a computing function doesn't fit.

What if the search is still too slow?

Divide the DAG into sub-graphs before search.

Evaluation

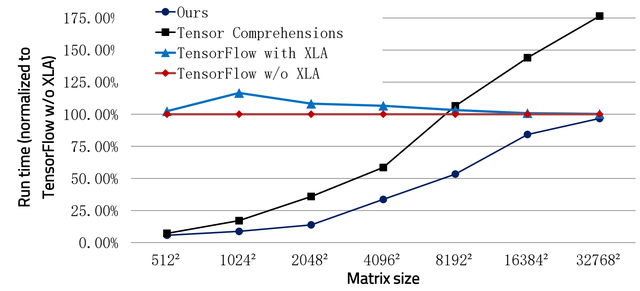

fig. 4. Fully Connected Layers normalized performance.The layer consists of a matrix multiplication, a bias and a ReLU. Batch size = 1.

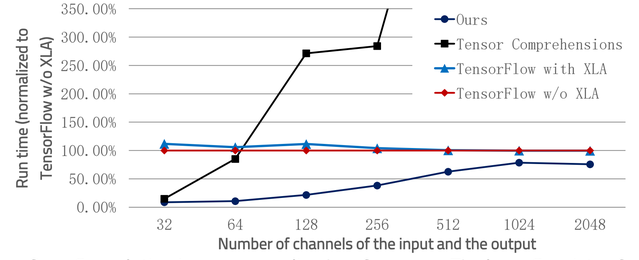

fig. 5. Convolution Layers normalized performance. The layer Consists of a convolution, a bias and a ReLU. Input heights and widths = 56. Batch size = 1.

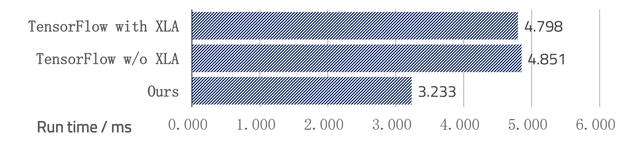

fig. 6. MobileNets Performance. No data for Tensor Comprehensions because it can't compute padded convolutions. Batch size = 1.

Environment: GPU server with 1 NVIDIA Tesla P100 GPU and dual Intel Xeon E5-2670 CPUs @ 2.30GHz. All numbers are average from >100 runs, and the first run is dropped to avoid lazy-initialization time.

Tested workload: Fully Connected Layers (*fig.* 4), Convolution Layers (*fig.* 5) and MobileNets (*fig.* 6).