TASO: Optimizing Deep Learning with Automatic Generation of Graph Substitutions

Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, and Alex Aiken

Stanford University

SOSP’19
Current Rule-based DNN Optimizations

Computation Graph

Rule-based Optimizer

Optimized Graph
Current Rule-based DNN Optimizations

TensorFlow currently includes ~200 rules (~53,000 LOC)

Fuse conv + relu

Fuse conv + batch normalization

Fuse multi. convs

Rule-based Optimizer
Limitations of Rule-based Optimizations

Robustness
Experts’ heuristics do not apply to all DNNs/hardware

Horovod with XLA is slower than without XLA (Tensorflow 1.12) #713

LweiiPeng opened this issue on Dec 19, 2018 - 2 comments

I have a distributed mrt model (Transformer-based, AdamOptimizer) using Horovod (0.15.1). When I turned on XLA under tensorflow 1.12, the training speed is about 20% slower instead of faster.

This result is sampled after training 1.5-hours and 4000 steps. I am using 4 V100 GPUs for the training.

Because my current software is tightly coupled with Horovod, I couldn’t test whether this is Horovod related or not.

Does anyone have experience on whether this is expected?

LweiiPeng commented on Dec 19, 2018

I added the question label on Dec 19, 2018

When I turned on XLA (TensorFlow’s graph optimizer), the training speed is about 20% slower.

With XLA, my program is almost 2x slower than without XLA.
Limitations of Rule-based Optimizations

**Robustness**
Experts’ heuristics do not apply to all DNNs/hardware

**Scalability**
New operators and graph structures require more rules

TensorFlow currently uses \(~4K\) LOC to optimize convolution
Limitations of Rule-based Optimizations

**Robustness**
Experts’ heuristics do not apply to all DNNs/hardware

**Scalability**
New operators and graph structures require more rules

**Performance**
Miss subtle optimizations for specific DNNs/hardware
The final graph is 30% faster on V100 but 10% slower on K80.
How should we address the complexity of designing DNN graph optimizations?
TASO: Tensor Algebra SuperOptimizer

- Key idea: replace manually-designed graph optimizations with automated generation and verification of graph substitutions for deep learning

- Less engineering effort: 53,000 LOC for manual graph optimizations in TensorFlow → 1,400 LOC in TASO

- Better performance: outperform existing optimizers by up to 2.8x
Graph Substitution

\[ Y_1 \xrightarrow{\text{Conv3x3}} W_1 \xrightarrow{X} W_2 \xrightarrow{\text{Conv3x3}} Y_2 \]

\[ Y_1 \xrightarrow{\text{Split}} W_1 \xrightarrow{\text{Concat}} W_2 \xrightarrow{X} Y_2 \]
TASO Workflow

Operator Specifications

Graph Subst. Generator

Candidate Substitutions

Graph Subst. Verifier

Verified Substitutions

Graph Optimizer
TASO Workflow

Input Comp. Graph → Search-Based Graph Optimizer → Optimized Comp. Graph

Verified Substitutions
Key Challenges

1. How to generate potential substitutions?
   Graph fingerprints

2. How to verify their correctness?
   Operator specifications + theorem prover
Graph Substitution Generator

Operators supported by hardware backend

Enumerate all possible graphs up to a fixed size using available operators
Graph Substitution Generator

66M graphs with up to 4 operators

Directly evaluating all pairs requires a quadratic number of tests.
Graph Substitution Generator

Compute output fingerprints with random input tensors
Pairs of graphs with identical fingerprint are candidate substitutions
Graph Substitution Generator

TASO generates ~29,000 substitutions by enumerating graphs w/ up to 4 operators

743 substitutions remain after applying pruning techniques to eliminate redundancy
Graph Substitution Verifier

Candidate Substitutions

Graph Subst. Verifier

Verified Substitutions

P1. conv is distributive over concatenation
P2. conv is bilinear
... Pn.

∀x, w₁, w₂ .
Conv(x, Concat(w₁, w₂)) =
Concat(Conv(x, w₁), Conv(x, w₂))

Operator Specifications
Verification Workflow

\[ \exists x, w_1, w_2. \]
\[ \left( \text{Conv}(x, w_1), \text{Conv}(x, w_2) \right) \neq \text{Split} \left( \text{Conv}(x, \text{Concat}(w_1, w_2)) \right) \]

P1. \( \forall x, w_1, w_2. \)
\[ \text{Conv}(x, \text{Concat}(w_1, w_2)) = \text{Concat}(\text{Conv}(x, w_1), \text{Conv}(x, w_2)) \]

P2. …

Operator Specifications

Theorem Prover

UNSAT
TASO generates all \textbf{743} substitutions in 5 minutes, and verifies them against \textbf{43} operator properties in 10 minutes.

Supporting a new operator requires \textbf{a few hours} of human effort to discover its properties.

Operator specifications in TASO $\approx \mathbf{1,400}$ LOC

Manual graph optimizations in TensorFlow $\approx \mathbf{53,000}$ LOC
Search-Based Graph Optimizer

- **Goal**: applying verified substitutions to obtain an optimized graph

- **Cost model**
  - Based on the sum of individual operators’ cost
  - Measure the cost of each operator on hardware

- **Cost-based backtracking search**
  - Backtrack local optimal solutions
  - Optimizing a DNN model takes less than 10 minutes

1. Z. Jia et al. Optimizing DNN Computation with Relaxed Graph Substitutions. In SysML’19.
End-to-end Inference Performance (V100 GPU w/ cuDNN)

- **ResNet-50**: 1.0x
  - TensorFlow
  - TensorRT
  - MetaFlow
  - TASO w/ cuDNN

- **NasNet-A**: 1.3x
  - TensorFlow
  - TensorRT
  - MetaFlow
  - TASO w/ cuDNN

- **ResNeXt-50**: 2.8x
  - TensorFlow
  - TensorRT
  - MetaFlow
  - TASO w/ cuDNN

- **NasRNN**: 1.4x
  - TensorFlow
  - TensorRT
  - MetaFlow
  - TASO w/ cuDNN

- **BERT-Large**: 1.4x
  - TensorFlow
  - TensorRT
  - MetaFlow
  - TASO w/ cuDNN

**Competition on standard models**

**Larger speedups on emerging models**
End-to-end Inference Performance (V100 GPU w/ TVM)

Similar speedups on the TVM backend
Different DNN models require **different** substitutions.
Conclusion

TASO is the first DNN optimizer that automatically generates substitutions

• Less engineering effort
• Better performance
• Formal verification

https://github.com/jiazhihao/taso

• Support DNN models in ONNX, TensorFlow, and PyTorch
Scalability Analysis

![Graph showing relative speedup vs maximum graph substitution size for NasNet-A, ResNeXt-50, and BERT.](image)
Case Study: NASNet

Add: element-wise addition
Conv: standard conv
DWC: depth-wise conv
Future Work: Query Optimizations

- A database query is expressed as a tree of relational operators
- Query optimizations are tree transformations
Contribution

• Replacing current manually-designed graph optimizations with *automatic generation* of graph substitutions for deep learning

• **Less engineering effort:** 53,000 LOC for graph optimizations in TensorFlow $\rightarrow$ 1,400 LOC

• **Better performance:** outperform existing optimizers by up to 2.8x

• **Correctness:** formal verification of graph substitutions
Limitations of Rule-based Optimizations

**Robustness**
Experts’ heuristics do not apply to all DNNs/hardware

**Scalability**
New operators and graph structures require more rules

**Performance**
Miss subtle optimizations for specific DNNs/hardware

Only apply to **specific hardware**

Only apply to **specialized graph structures**
TASO: Tensor Algebra SuperOptimizer

Key idea: automatically \textit{generate} graph substitutions and \textit{verify} them
TASO: Tensor Algebra SuperOptimizer

Input Comp. Graph

Operator Specifications

Search-Based Graph Optimizer

Graph Subst. Verifier

Graph Subst. Generator

Verified Substitutions

Candidate Substitutions

TASO
End-to-end Inference Performance

Relative Speedup over Existing Frameworks

- TensorFlow
- TensorRT
- MetaFlow
- TASO

- ResNet-50: 1.0x
- NasNet-A: 1.3x
- ResNeXt-50: 2.8x
- NasRNN: 1.4x
- BERT-Large: 1.4x
Joint Optimizer for Graph Substitution and Data Layout

- **Motivation**: some graph substitutions only improve performance when combined with particular layout transformations

- **Idea**: consider potential layout transformations along with graph substitutions (additional 1.3x speedup)

- **Cost-based backtracking search**
  - Assume the cost to run a model is the sum of individual operators’ costs
  - Measure the cost of each operator on hardware
  - A search takes less than **10 minutes**